UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

Aplicación del Método de Sensado Comprimido: Cámara de un Solo Sensor

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Físico

PRESENTA:

Andrés Rello Rincón

TUTOR

Dr. Pedro Antonio Quinto Su

Ciudad de México 2021

1. Datos del Alumno
Rello
Rincón
Andres
5560662093
Universidad Nacional Autónoma de México
Facultad de Ciencias
Física
311686777

2. Datos del tutor
Dr
Pedro Antonio
Quinto
Su

3. Datos del sinodal 1
Dr.
Arnulfo
Martínez
Dávalos

4. Datos del sinodal 2
Dr
Carlos
Villareal
Luján

5. Datos del sinodal 3
M. en I.A.
José Luis
Jiménez
Andrade

6. Datos del sinodal 4
Dr
Gibran
Fuentes
Pineda

7.Datos del trabajo escrito
Aplicación del Método de Sensado Comprimido: Cámara de un Solo Sensor
94 p
2021

Agradecimientos

Quiero agradecer al Dr. Pedro A. Quinto por todo el apoyo que me brindó para poder desarrollar y concluir este trabajo. Me deja grandes enseñanzas las cuales estoy seguro serán vitales para mi futuro desarrollo.
A mis padres y hermano les quiero dar las gracias por siempre apoyarme y entenderme. Este logro es en gran parte suyo también.
A mi abuela le agradezco el abrirme las puertas de su casa durante mis años de estudio y por la gran compañera que ha sido para mí.
Finalmente quiero agradecer al resto de mi familia, mexicana y colombiana, y a mis amigos.

II

Índice general

Capítulo 1

Introducción

Una de las formas más usadas para obtener información sobre el mundo que nos rodea es a través de señales. Estas han permitido el entendimiento de infinidad de fenómenos y hoy en día son utilizadas ampliamente en diferentes áreas como telecomunicaciones y medicina. Es por esto que se han convertido en un amplio campo de estudio. Una de las áreas de desarrollo de este campo, la cual es tema central de esta tesis, es la adquisición de señales. En esta área ha habido avances que hoy en día permiten transmitir grandes cantidades de información por todo el planeta e incluso fuera de este, siendo un ejemplo ampliamente usando el sistema de posicionamiento global o GPS por sus siglas en inglés [19]. Muchos de estos avances se han dado en el desarrollo de dispositivos más eficientes y de mayor precisión, sin embargo algunos se han enfocado más en la estructura de las señales en sí, para intentar sacar ventaja de estas. Este es el caso del sensado comprimido o CS (por sus siglas en inglés), un método de adquisición de señales desarrollado en décadas reciente. Este método tiene dos características principales que lo distinguen de los demás. La primera es que no obtiene la señal completa, sino una versión comprimida de esta y la segunda es que la adquisición no es adaptativa, lo que significa que el funcionamiento del método es igual para cualquier señal. Posteriormente a su adquisición la señal completa se puede obtener con algoritmos de reconstrucción a partir de la señal comprimida.

En este trabajo se busca dar una demostración de una de las muchas aplicaciones que puede tener el CS. Esta, en específico, es en al área de imágenes

digitales. Para entender esta aplicación primero hay que mencionar que las cámaras digitales comúnmente utilizadas cuentan con un sensor por cada pixel de la fotografía que obtienen, por ejemplo si se tienen fotos de 1 megapixel (MP) esto quiere decir que la cámara cuenta mínimo con 1 millón de sensores. Teniendo esto en cuenta, el principal objetivo de este trabajo consiste en construir un dispositivo parecido al que se presenta en [11], que se puede describir como una cámara digital capaz de obtener fotografías utilizando únicamente un sensor. La forma lógica de lograr esto es medir un pixel a la vez usando el único sensor que se tiene, de esta manera si se quiere obtener una imagen de 1MP se tienen que realizar 1 millón de mediciones. Sin embargo si se aplica el método de sensado comprimido, lo cual conforma la segunda parte del objetivo del trabajo, este número se puede reducir drásticamente. En esta tesis se trabaja con una sola dimensión de imagen que es 32x32 lo que equivale a fotografías digitales de 1024 pixeles. La dimensión seleccionada es pequeña debido a que los algoritmos de reconstrucción son computacionalmente muy costosos y para resoluciones superiores los tiempos de cálculo pueden aumentar mucho.

Como se menciona al final del párrafo anterior muchos de los algoritmos que se usan para la reconstrucción de las señales comprimidas en el CS son computacionalmente muy costosos y complejos. En años recientes se han usado redes neuronales artificiales (RNAs), que son modelos computacionales que tienen la capacidad de aprender alguna tarea específica, para sustituir estos algoritmos. Las ventajas que tienen las RNAs es que por lo general son fáciles de implementar y los tiempos de computo se pueden acelerar usando unidades de procesamiento gráfico (GPUs) o unidades de procesamiento tensorial (TPUs). Por estas razones en esta tesis se opta por el uso de RNAs aplicadas al CS. En total se usan 5 diferentes arquitecturas, las cuales se obtienen directa o indirectamente de [14], [11] y [20] y a lo largo del trabajo se comparan entre sí para ver cual da el mejor desempeño.

Esta tesis está dividida en 6 capítulos. El primero corresponde a la introducción donde se da una idea general sobre el contenido de la misma. El segundo desarrolla a detalle la teoría del sensado comprimido. El tercer capítulo contiene los fundamentos teóricos generales de las redes neuronales artificiales, ahonda un poco más en los puntos particulares que se tocan con los modelos y describe cómo se pueden aplicar al sensado comprimido. En el

cuarto capítulo se describen los modelos utilizados, y como se entrenaron. En el penúltimo se explica a detalle el montaje experimental, su funcionamiento y las mediciones que se realizan con él. Además se presentan las fotografías finales obtenidas y se comparan los modelos desde un punto de vista visual. En el sexto capítulo se analizan y comparan los desempeños de los modelos con imágenes digitales. También se estudia su respuesta ante señales con ruido. Finalmente en el último capítulo se comentan una serie de conclusiones sobre el presente trabajo.

Capítulo 2

Sensado Comprimido

Con el desarrollo de las tecnologías de señales, el campo de adquisición y procesamiento de estas ha obtenido mucha relevancia. Dentro de este campo hay diferentes retos, uno de los cuales es el muestreo de señales cuando el proceso es muy costoso o cuando se tienen ciertos impedimentos para obtener la señal completa. Dicho en otras palabras, el objetivo es como obtener toda la información de una señal sin tener que medirla completamente. Actualmente el enfoque convencional para solucionar este problema usa como base el teorema de Nyquist-Shannon [16] que se enuncia a continuación:

Teorema 1. *Una señal f con un ancho de banda limitado, es decir que no contiene frecuencias mayores a una cierta frecuencia Ω_N, se puede representar de forma única a partir de mediciones periódicas tomadas a una frecuencia $\Omega_s > 2\Omega_N$.*
A Ω_N se le conoce como la frecuencia de Nyquist.

Otra manera de interpretar este teorema es que dada una señal existe una cota inferior para el número de mediciones (periódicas) que se necesitan hacer para poderla reconstruir posteriormente. A continuación se plantea de manera matemática el muestreo de una señal, que a partir de ahora se supondrá discreta ya que esto simplificará el desarrollo teórico en secciones posteriores y además es el tipo de señal con el que se trabaja en es esta tesis.

Dada una señal $\vec{f} \in \mathbb{R}^n$ el muestreo se realiza a partir del producto punto de ella y un conjunto de elementos $\{\vec{\varphi}_i\}_{i=1}^m$ de $\mathbb{R}^n$. De esta forma la informa-

ción que se obtiene de la señal $\vec{f}$ queda contenida en el vector $\vec{y} \in \mathbb{R}^m$ cuya i-esima entrada está definida como:

$$y_i = \langle \vec{f}, \vec{\varphi}_i \rangle \tag{2.1}$$

Donde $\langle\ ,\ \rangle$ representa el producto punto.

Para dejar clara la idea anteriormente expuesta y cómo se relaciona con este trabajo tómese la señal $\vec{f}$ como una imagen que se quiere representar con 1024 (32x32) pixeles. Esto quiere decir que el espacio en el que se está trabajando es $\mathbb{R}^{1024}$. El caso más común de muestreo es el que realizan las cámaras tradicionales en donde el conjunto $\{\vec{\varphi}_i\}_{i=1}^m$ es la base canónica, en este caso $m = n = 1024$. Lo que se tiene muestreando de esta forma es lo siguiente:

$$\vec{y} = \sum_{i=1}^{n} \langle \vec{f}, \vec{\varphi}_i \rangle = \sum_{i=1}^{n} f_i \vec{\varphi}_i = \vec{f} \tag{2.2}$$

Que en resumen significa que se muestrea por separado la información de cada pixel, por lo que $\vec{y}$ representa directamente la imagen que se desea obtener. Lo que se hace físicamente es tomar la información correspondiente al primer pixel con un sólo sensor. Después realizar lo mismo para el segundo y así sucesivamente. Dada la frecuencia de Nyquist de la señal en cuestión se puede obtener la frecuencia mínima para realizar mediciones asegurando una correcta representación de la señal o visto de otra forma existe un conjunto $\{\vec{\varphi_i'}\}_{i=1}^{m'}$ con $m' \leq 1024$ que permite muestrear la imagen (señal) y posteriormente recuperarla. Entre más chica sea la frecuencia de Nyquist menor será m' sin embargo es claro que tiene una cota inferior. La pregunta interesante en este caso es ¿Qué pasa cuando el número de elementos del muestreo es menor que el límite impuesto por el teorema de Nyquist-Shannon? Es aquí donde entra en juego el sensado comprimido ya que permite, con una probabilidad muy alta y bajo ciertas condiciones que se mencionan más adelante, reconstruir la señal original con un número de mediciones inferior a este.

En el párrafo anterior se comentó que cuando se obtiene una imagen en su forma no comprimida es muy común usar como conjunto de muestreo la

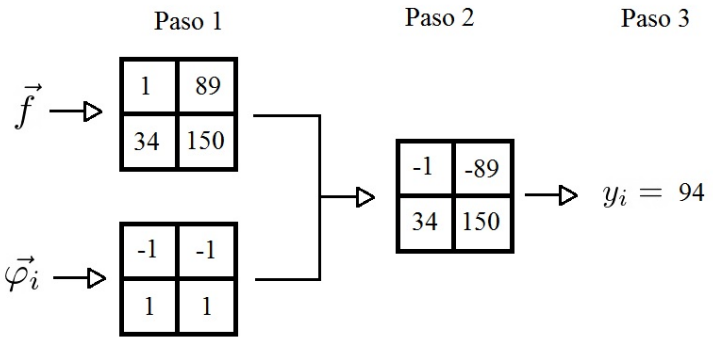


Figura 2.1: Visualización gráfica del proceso de sensado de una imagen de 4 pixeles ($\vec{f}$) con un elemento $\vec{\varphi}_i$ construido con elementos 1 y -1. El valor final del muestreo correspondiente es equivalente al producto punto $\langle \vec{f}, \vec{\varphi}_i \rangle$ y corresponde a la entrada y_i del vector de muestreo $\vec{y}$.

base canónica en el espacio de pixeles. Sin embargo cuando la información se quiere obtener en forma comprimida la estructura del conjunto de muestreo suele cambiar. En la figura 2.1 se encuentra una representación gráfica de cómo se haría el muestreo para uno de estos elementos. En ella, en el paso 1 se observa una imagen de 4 pixeles representada por $\vec{f}$ y el elemento $\vec{\varphi}_i$ con el que se realiza el muestreo. El paso 2 que representa la primera parte del producto punto que corresponde a multiplicar los elementos de $\vec{f}$ y $\vec{\varphi}_i$ entrada a entrada y finalmente en el paso 3 se suman los productos resultantes para obtener un solo valor que representa el muestreo final para el elemento en cuestión. Para obtener el muestreo total que se representa como el vector $\vec{y}$, este proceso se debe repetir para cada elemento del conjunto $\{\vec{\varphi}_i\}_{i=1}^{m'}$.

2.1. Bases del sensado Comprimido

El sensado comprimido busca recuperar una señal a partir de un muestreo no adaptivo de esta, el cual se encuentra descrito en la expresión 2.1. El término no adaptivo significa que el proceso de muestreo no depende de la señal particular con la que se trabaja, sino que es el mismo para todas. Antes de plantear en que consiste este, se describe un método más simple que busca resolver el mismo problema y que permite, de forma más clara, entender

cuáles son las ventajas que tiene.

El método que se presenta a continuación está descrito en [7], y puede ser usado para realizar una reconstrucción, no adaptiva, de la señal a partir de los datos del muestreo realizado. Hay que mencionar que el tipo de elemento φ_i utilizado para el muestreo puede influir bastante en el desempeño del método. Lo que se hace muchas veces es construir estos elementos a partir de alguna distribución aleatoria como puede ser una normal o una Bernoulli. Si se tiene entonces una señal $\vec{f} \in \mathbb{R}^n$ desconocida y un vector de muestreos $\vec{y}$ generado a partir de los elementos en el conjunto$\{\vec{\varphi}_i\}_{i=1}^m$, la reconstrucción de $\vec{f}$ se puede obtener a partir de la siguiente expresión:

$$\vec{f} \simeq \frac{1}{m} \sum_{i=1}^{m} (y_i - \langle \vec{y} \rangle) \varphi_i \tag{2.3}$$

Donde $\langle \vec{y} \rangle$ representa el promedio de los m muestreos realizados. Como se verá más adelante este método no genera buenos resultados a menos que m sea muy grande.

El sensado comprimido es otro método que tiene el mismo objetivo que el descrito en el párrafo anterior. Sin embargo realizando algunas suposiciones, las cuales se comentan más adelante, logra una calidad de reconstrucción muy superior. Para poder entender en que consiste hay que definir dos conceptos: Incoherencia y señales dispersas. El primero esta relacionado principalmente con el muestreo de la señal y el segundo con la reconstrucción.

2.1.1. Señales dispersas:

Tómese una señal cualquiera $\vec{f} \in \mathbb{R}^n$. Esta señal se puede representar de diferentes formas dependiendo de la base que se use para hacerlo. Dentro de todas estas bases del espacio $\mathbb{R}^n$ supóngase que existe una que es ortonormal, lo que no es necesario pero simplifica el desarrollo del tema, que se denotará como $B_\psi = \{\vec{\psi}_i\}_{i=1}^n$ con la cual se puede expresar a $\vec{f}$ de la siguiente manera:

$$\vec{f} = \Psi \vec{x} \tag{2.4}$$

Donde la columna i de la matriz Ψ es el elemento i de la base B_ψ y el vector $\vec{x}$ representa los coeficientes de cada elemento de la base.

Si en su representación bajo la base B_ψ una gran parte de las entradas de $\vec{x}$ son cero o muy pequeños a comparación el resto, entonces se considera a $\vec{f}$ como una señal dispersa. También se puede decir que cuando esto se cumple $\vec{f}$ es compresible, lo que quiere decir que su información se puede expresar en un espacio de dimensión menor al espacio al que pertenece.

La ventaja de las señales compresibles es que se pueden representar de una forma mucho más concisa. Esto es muy útil por ejemplo para la compresión de archivos digitales. En vez de guardar el archivo completo se pueden guardar los coeficientes significativos en su representación dispersa y de estos se puede obtener la señal original en cualquier momento.

2.1.2. Incoherencia:

Supóngase que se tiene un par de bases ortonormales de $\mathbb{R}^n$, nuevamente esto no es necesario pero simplifica varios conceptos, $B_\varphi = \{\vec{\varphi_i}\}_{i=1}^n$ y $B_\psi = \{\vec{\psi_i}\}_{i=1}^n$. La primera de estas sirve para llevar a cabo el muestreo de la señal y se llamará base de muestreo. La segunda se llamará base de representación y sirve para representar la señal $\vec{f}$ a partir de su vector de coeficientes $\vec{x}$. La coherencia entre estas dos bases se define como:

$$\mu(B_\varphi, B_\psi) = \sqrt{n} \max_{1 \leq k,j \leq n} |\langle \varphi_k, \psi_j \rangle| \tag{2.5}$$

Este escalar se puede entender como una medición de la correlación más grande entre 2 elementos de las dos bases y vive dentro del intervalo $[1, \sqrt{n}]$.

Para que el método de CS logre un desempeño óptimo se tiene que trabajar con pares de bases que tengan una coherencia muy baja. A continuación se intenta explicar de forma intuitiva porqué se necesita esta condición. Para esto supóngase nuevamente una cierta señal $\vec{f} \in \mathbb{R}^n$ que se puede expresar de forma dispersa con el vector de coeficientes $\vec{x}$ de la siguiente manera: $\vec{f} = \Psi\vec{x}$. Donde las columnas de la matriz Ψ corresponden a los elementos de la base

B_φ. El proceso de muestreo de la señal $\vec{f}$ se puede escribir entonces como $\vec{y} = A\vec{x}$ donde $A = \Phi\Psi$ y Φ es una matriz que tiene los elementos de la base B_φ como renglones. La matriz A se conoce como matriz de sensado. El primer punto importante que se tiene que acentuar es que se puede definir la coherencia de las bases B_φ y B_ψ a partir de los elementos de la matriz de sensado. De esta forma se tiene que $\mu(B_\varphi, B_\psi) = \mu(A)$ definiendo el segundo término de la siguiente manera:

$$\mu(A) = \sqrt{n} \max_{1 \leq k,j \leq n} |a_{i,j}| \tag{2.6}$$

Para ilustrar las consecuencias de la coherencia sobre la matriz de sensado hay que ver qué pasa cuando esta es máxima y se da el peor caso. Coherencia máxima entre las dos bases implica que existe al menos un par de enteros $k, l \in [1, N]$ tal que $|\langle \phi_k, \psi_l \rangle| = 1$ y por lo tanto $\langle \phi_k, \psi_i \rangle = \langle \phi_j, \psi_l \rangle = 0$ para todo $i \neq l$ y $j \neq k$ respectivamente, que dicho en otras palabras significa que el renglón k de A tiene ceros en todas sus entradas excepto en la l y la columna l tiene ceros en todas sus entradas excepto en la k. Ahora supóngase que no se obtiene la información total de la señal, esto se puede simular construyendo una nueva matriz de muestreo Φ_m conformada por m filas de la matriz Φ lo que es equivalente a seleccionar m elementos de la base B_ϕ con los que se hará el muestreo. Supóngase que la fila k de Φ no fue seleccionada para construir Φ_m. Esto implica que si se construye la nueva matriz de sensado $A_m = \Phi_m\Psi$ la columna l de A_m tendrá en todas sus entradas ceros y si resulta que la señal $\vec{f}$ en su representación bajo Ψ tiene un vector de coeficientes $\vec{x}$ tal que $x_i = 0$ para todo $i \neq l$ entonces $\vec{y} = A_m\vec{x} = \vec{0}$. Esto quiere decir que después del muestreo la información que se tiene sobre la señal es nula y en este caso no existe ningún método que pueda reconstruir la señal original.

En resumen el problema cuando la coherencia es grande es que un grupo reducido de elementos de la matriz de sensado toman valores grandes mientras que una mayoría se vuelven muy pequeños. Esto tiene como consecuencia, cuando no se muestrea la señal completa, que la probabilidad de que $\vec{x}$ esté en el kernel de A aumente y por lo tanto el problema no se pueda resolver.

2.2. Reconstrucción de la señal

A continuación se describe a grandes rasgos el proceso de CS haciendo una recapitulación de [2]. Primero se lleva a cabo el muestreo con elementos de las base B_φ. Supóngase que no se puede muestrear la señal completa que, como ya se mencionó, equivale a usar una matriz Φ_m definida en la sección anterior. El muestreo se puede escribir entonces como:

$$\vec{y} = \Phi_m \vec{f}, \qquad \text{donde } \vec{y} \in \mathbb{R}^m \tag{2.7}$$

El siguiente paso es la reconstrucción de la señal original. Para llevar a cabo esto se cuenta con la información contenida en $\vec{y}$ y se supone que $\vec{x}$ es disperso, donde $\vec{f} = \Psi\vec{x}$. Usando esto, la señal puede ser recuperada con el siguiente programa de optimización convexo:

$$\min_{\vec{x'} \in \mathbb{R}^n} |\vec{x'}|_{l_1}, \qquad \text{dado que } \vec{y} = \Phi_m \Psi \vec{x'} \tag{2.8}$$

Expresado en palabras, se está buscando un vector de coeficientes $\vec{x'}$ que sea consistente con las mediciones obtenidas y que además tenga una norma l_1 mínima. Esto último viene de suponer que la señal $\vec{f}$ es dispersa bajo la base B_ψ.

Una pregunta que puede surgir una vez planteado 2.8 es si en el caso de CS existe una cota inferior para m o este puede ser tan chico como se desee. Un primer resultado, obtenido de [2], relacionado a esto se enuncia a continuación:

Teorema 2. *Dada una señal $\vec{f} \in \mathbb{R}^n$ y suponiendo que su vector de coeficientes $\vec{x}$ bajo la base B_ψ tiene a lo más S entradas diferentes de cero, se realiza un muestreo aleatorio con m elementos de la base B_φ. Si*

$$m \geq C \cdot \mu^2(B_\varphi, B_\psi) \cdot S \cdot log(n)$$

para alguna constante positiva C, entonces la solución de 2.8 es exacta con una probabilidad muy alta.

En el Teorema 2 se puede ver claramente el efecto de la coherencia con relación al número de muestras que se necesitan obtener de la señal. Si la coherencia es igual a 1, entonces el número de muestras que se necesitan es del orden de $S \cdot log(n)$ que puede llegar a ser mucho menor que n. Otro punto importante que se puede obtener del teorema es que en CS el muestreo no depende de la señal. No importa de que elementos de la base B_ϕ esté conformado el muestreo, la probabilidad de no recuperar la señal original es muy baja. Y cuando m es lo suficiente grande, a efectos prácticos esta probabilidad es prácticamente cero.

El Teorema 2 permite estimar a m para aplicar el método de CS sobre una señal perfectamente dispersa en el dominio de B_ψ. Sin embargo la representación de señales reales no es en general perfectamente dispersa. Por esta razón hay que detenerse a pensar si el método de CS es lo suficientemente robusto para funcionar con este tipo de señales. Para esto hay que introducir primero un concepto que se conoce como "Restricted Isometry Property" o RIP y se define en [2]:

Definición 1. *Para cada entero $S = 1, 2, ...$ se define la constante de isometría δ_S de una matriz A como el menor número que cumple con:*

$$(1 - \delta_S) \cdot |\vec{x}\|_{l_2}^2 \leq |A\vec{x}\|_{l_2}^2 \leq (1 + \delta_S) \cdot |\vec{x}\|_{l_2}^2$$

Para cualquier vector $\vec{x}$ con a lo más S entradas diferentes de cero.

Se dice que una matriz A cumple con RIP de orden S cuando su constante de isometría δ_S no es muy cercana a uno.

Si se recuerda la discusión en la sección 2.1.2 el problema con pares de bases con coherencias altas es que el vector $\vec{x}$ de coeficientes de la señal $\vec{f}$ podía fácilmente pertenecer al kernel de $A_m = \Phi_m \Psi$ y en este caso es imposible reconstruir la señal. Teniendo esto en cuenta es fácil ver la utilidad de RIP. Una matriz A que la cumple garantiza que señales dispersas con a lo más S entradas diferentes de cero no estará en su kernel.

Teorema 3. *Supóngase que $\delta_{2S} < \sqrt{2} - 1$. Entonces la solución a 2.8 cumple con lo siguiente:*

$$|\vec{x'} - \vec{x}|_{l_1} \leq C_0 \cdot |\vec{x} - \vec{x_s}|_{l_1}$$

Para alguna constante positiva C_0 y donde $\vec{x_S}$ es el vector $\vec{x}$ con todas sus entradas igualadas a cero excepto las S mayores.

Mientras se respeten los supuestos del Teorema 3 se garantiza una reconstrucción perfecta si $\vec{x} = \vec{x_S}$ y cuando $\vec{x}$ no es perfectamente disperso se garantiza que se obtendrá la información de las S entradas más significativas de $\vec{x}$. Hay que mencionar que hay resultados muy parecidos a los del teorema 2 cuando los muestreos tienen ruido, haciendo de esta forma el senado comprimido robusto ante estos escenarios. Estos resultados se pueden encontrar en [2].

Lo último que falta por mencionar es como se relacionan S, que representa el número de entradas que se pueden recuperar del vector de coeficientes $\vec{x}$, y m, que indica el número de mediciones que se realizarán. En [2] se mencionan diferentes matrices de sensado que cumplen con RIP de orden S y dan su relación con m. Aquí se menciona una de estas que es la que resulta relevante para este trabajo como se verá más adelante. Suponiendo que se tiene una base de representación fija B_ψ, si se construye la matriz de muestreo Φ_m llenando cada entrada de forma independiente con una distribución normal o una Bernoulli simétrica entonces la matriz $A = \Phi_m \Psi$ cumplirá con RIP de orden S con una probabilidad muy grande mientras se cumpla:

$$m \geq C \cdot S \cdot log(\frac{n}{S}) \tag{2.9}$$

Hay que tener en cuenta que durante muchos años los resultados en este campo generaban mucha controversia debido a que no había sustento teórico para justificarlos. No fue hasta hace relativamente poco tiempo que resultados rigurosos, algunos de los cuales se mencionaron anteriormente, han sido planteados. Esto ha hecho que el campo del sensado comprimido tome relevancia y más gente lo empiece a tomar en cuenta para diferentes aplicaciones.

Capítulo 3

Redes Neuronales Artificiales (RNAs)

El estudio del cerebro siempre ha sido de gran interés para muchas áreas de la ciencia. Una de las razones es que es la estructura biológica que ha permitido al ser humano el gran desarrollo tecnológico con el que hoy cuenta. En este sentido el poder replicar su estructura y funcionamiento podría permitir crear programas para solucionar diferentes problemas de una forma similar a como lo hacen los humanos.

A mediados del siglo pasado se lograron grandes avances en esta área. En 1943 W. McCulloch y W. Pitts introducen la idea de redes neuronales como máquinas de computo [13] y en 1949 D. Hebb postula un mecanismo básico de plasticidad sináptica, que es la forma en la que el cerebro retiene conocimiento a partir de las sinapsis donde se comunican las neuronas. En 1958 F. Rosenblatt propuso el perceptrón [18], un modelo artificial que puede aprender alguna tarea de clasificación a partir de ejemplos de esta. A esto se le conoce como aprendizaje supervisado. La contribución de este último es esencial para comprender el funcionamiento actual de las redes neuronales artificiales.

Hoy en día las RNAs han mostrado una gran versatilidad y han sido aplicadas en diferentes áreas con gran éxito. Una de las áreas donde su desempeño ha sido extraordinario es en tareas relacionadas con el manejo y procesamien-

to de imágenes digitales. Particularmente han sido aplicadas al método de CS ([14], [11], [20]) y en este trabajo se sigue este camino.

En este capítulo se empieza con una pequeña descripción del perceptrón y posteriormente se desarrollan los puntos más importantes de las RNAs, además se explica un poco más a profundidad los puntos más relevantes que conciernen a este trabajo. Al final se explica la forma en la que las RNAs pueden ser aplicadas al CS y las ventajas y desventajas de hacerlo.

3.1. Perceptrón

El objetivo del perceptrón es replicar, de forma muy simple, el funcionamiento de una neurona. Este está compuesto por una serie de neuronas artificiales, cada una de las cuales representa la unidad básica de procesamiento. En esta tesis el término neurona hará referencia a la neurona artificial antes mencionada. El perceptrón cuenta con una capa de entrada y otra de salida, cada una formada por un cierto número de neuronas artificiales. Cada neurona en la capa de entrada recibe un dato y lo transmite a cada neurona de la capa de salida pesado con un cierto parámetro. Por su parte, una neurona de la capa de salida recibe los datos pesados por los parámetros de la capa de entrada, los suma junto con un parámetro llamado bias y al resultado le aplica una función específica que se conoce como función de activación. Hay que mencionar que los parámetros pueden tomar cualquier valor en $\mathbb{R}$. Para entender mejor esto, a continuación se describe el funcionamiento de forma más formal, suponiendo que únicamente se tiene una neurona en la capa de salida para simplificar la notación. En la figura 3.1 se tiene una representación de un perceptrón con 2 neuronas en la capa de entrada y una en la de salida que se puede usar para guiar la explicación. Supóngase primero un perceptrón que tiene N neuronas en la capa de entrada, esto quiere decir que recibirá N datos: $x_1, x_2, ..., x_N$ como entrada. Por cada dato de entrada x_i el perceptrón tiene definido un parámetro w_i y si se expresan los datos de entrada como un vector $\vec{x}$ y sus parámetros correspondientes como un vector $\vec{w}$ entonces se puede expresar el resultado y de la neurona artificial de la capa de salida como:

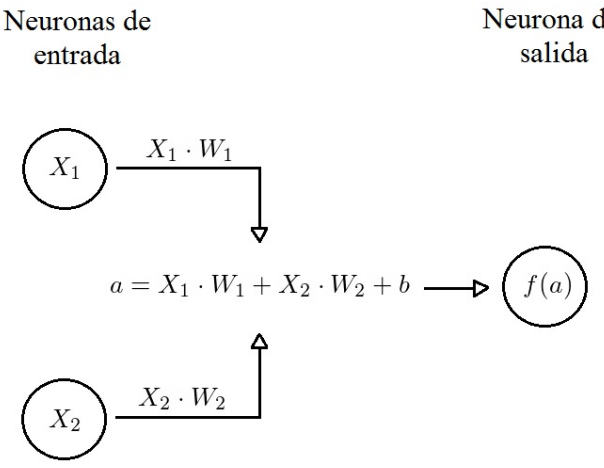


Figura 3.1: Digrama representativo de un perceptrón con 2 neuronas en la capa de entrada y 1 en la capa de salida. Los datos de entrada estan denotados como X_1 y X_2, los pesos correspondientes como W_1 y W_2 y el bias como b.

$$y = f(\vec{w} \cdot \vec{x} + b) \tag{3.1}$$

Donde f es la función de activación y en el caso del perceptrón está definida como:

$$f(a) = \begin{cases} 1 \text{ si } a \geq 0 \\ 0 \text{ si } a < 0 \end{cases} \tag{3.2}$$

El término b es el parámetro bias anteriormente mencionado y determina que tan fácil se obtiene del perceptrón un 1 como resultado.

Un perceptrón sirve para realizar clasificación de datos en clases, lo cual tiene muchas aplicaciones. Un ejemplo es un perceptrón con dos neuronas en la capa de entrada y una en la capa de salida, como el que se encuentra en la figura 3.1, que es capaz de representar relaciones lógicas como AND o OR [17]. Por ejemplo para la relación AND se pueden hacer $W_1 = W_2 = W > 0$ y $b = -\frac{3W}{2}$ y se puede comprobar que la tabla lógica de AND se cumple. Para modelos con pocas neuronas como los ya mencionados, no es

complicado hallar manualmente los valores de los parámetros, sin embargo si el número de neuronas crece, esta tarea puede llegar a ser muy complicada. La solución a este problema llegó con el desarrollo de algoritmos de aprendizaje que permitieron al perceptrón ajustar de forma automática sus parámetros. En la sección 3.2.4 se discuten estos algoritmos. Un gran problema que hizo perder por un tiempo el interés en el campo fue que el perceptrón no era capaz de realizar clasificación en conjuntos de objetos que no fueran linealmente separables, por ejemplo la relación XOR [17]. Sin embargo se hayó una solución al problema que fue la creación del perceptrón multicapas o MLP, por sus siglas en inglés, el cual además de la capa de entrada y salida, cuenta con un cierto número de capas intermedias. Este avance tomó tiempo ya que se tuvieron que desarrollar nuevos algoritmos de aprendizaje que funcionaran con la arquitectura de multicapa, pero sentó las bases de las RNAs modernas que hasta hoy han logrado grandes avances en diferentes campos.

3.2. Redes Neuronales Artificiales (RNAs)

Se toma como definición de RNA la que hace Haykin en [4]: Una red neuronal artificial es un procesador distribuido paralelo conformado por unidades de procesamiento simples que tienen una tendencia natural a retener conocimiento experimental y hacerlo disponible para su uso. Las RNA simulan al cerebro de dos formas:

- El conocimiento es adquirido por la RNA de su entorno a través de un proceso de aprendizaje
- El conocimiento se guarda en la fuerza de las conexiones interneuronales, denominadas pesos sinápticos, que están representados por los parámetros.

Hoy en día las RNAs están inspiradas en el perceptrón multicapa. Las principales diferencias se encuentran en el tipo de capas, de los cuales se mencionan algunos más adelante y de modificaciones estructurales en las neuronas que en conjunto han permitido a las RNAs resolver problemas cada vez más complejos.

3.2.1. Clasificación de RNAs

Existen diferentes formas de clasificar las RNAs. Aquí se usa una muy general, en función de su arquitectura, para dividirlas en dos clases [4]:

- Redes neuronales prealimentadas (RNP): En este tipo de RNAs las entradas de la neurona artificial de una capa especifica son normalmente las salidas de las neuronas artificiales de la capa anterior. De forma un poco más formal las entradas de una neurona en la capa i jamás pueden venir de una capa $j \geq i$.
- Redes neuronales recurrentes (RNR): A diferencia de las RNP en este caso las neuronas de una capa i pueden recibir entradas que vengan de una capa j, incluso si $j \geq i$

.

En este trabajo se utilizan únicamente RNPs. Por esta razón se profundiza en la teoría únicamente en este tipo de RNAs

3.2.2. Funciones de activación

En las RNAs la unidad básica de procesamiento es una neurona artificial que tiene la misma estructura que las neuronas del perceptrón con un pequeño cambio que tiene grandes consecuencias. Este cambio radica en la función de activación. Como se mencionó el perceptrón usa 3.2 como función de activación. Esta función en específico tiene un problema con los algoritmos modernos de aprendizaje, los cuales se mencionan en la sección 3.2.4, y es que su derivada es siempre cero y esto no permite el aprendizaje. Por esta razón con el tiempo empezaron a surgir nuevas funciones de activación. En esta sección se mencionan algunas de las más usadas:

funcion sigmoide:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.3}$$

Funcion ReLU:

$$R(z) = \max(0, z) \tag{3.4}$$

Existen muchas más funciones de activación, cada una de las cuales tiene ventajas y desventajas. Algunas de estas se pueden revisar en [3].

3.2.3. Tipos de capas

Como ya se ha mencionado las RNAs se pueden entender como capas de neuronas interconectadas. Hoy en día existen muchos tipos de capas, cada una de las cuales funciona de forma diferente y sirve para diversos propósitos. A continuación se hace la descripción de 3 tipos de capas, que son los que se usaron en el desarrollo de este trabajo. Estos tres tipos de capas se usan con RNPs, sin embargo no son exclusivas de este tipo de RNAs.

Capa totalmente conectada:

Este tipo de capa es el más simple conceptualmente y su nombre es bastante explícito. Una capa totalmente conectada está caracterizada por tener cada una de sus neuronas conectada con todas las neuronas de la capa anterior. Y cuando se dice que está conectada se refiere a que recibe como entradas el valor de salida de cada una de las neuronas de las capa anterior. Se puede observar una red con dos capas totalmente conectadas en la figura 3.2.

Capa convolucional:

La siguiente explicación para describir el funcionamiento de este tipo de capa se basa en el capítulo 6 de [15]. Hasta ahora las entradas de las capas de una RNA se han pensado como una línea de neuronas, en este caso, para poder simplificar la explicación, se pensarán como arreglos cuadrados de neuronas (esto se puede generalizar a más dimensiones para otro tipo de datos). Supongamos entonces que la RNA tiene como capa de entrada un arreglo de 28x28 neuronas. A diferencia de una capa totalmente conectada, una neurona de la capa convolucional no recibirá información de todas las neuronas de la capa anterior sino de un grupo reducido de vecinos. El tamaño del grupo de

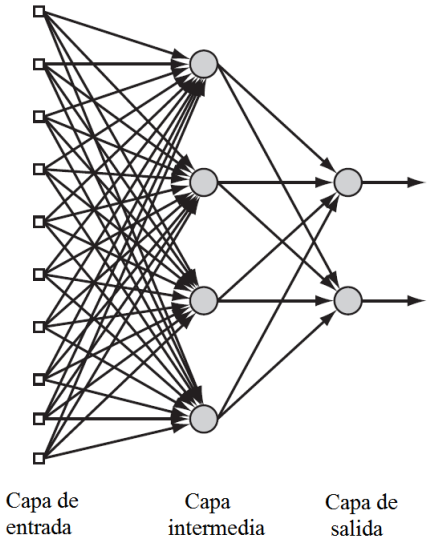


Figura 3.2: RNA totalmente conectada [4]

vecinos estará definido por el tamaño de un kernel. Para seguir el ejemplo en [15] digamos que este kernel es de 5x5, esto quiere decir que una neurona de la capa convolucional recibirá únicamente información de una vecindad de 25 neuronas de la capa anterior como se observa en la figura 3.3. La región de la cual recibe información cada neurona de la capa convolucional se genera desplazando el kernel sobre el arreglo de entrada. En este ejemplo, si se empieza poniendo el kernel en la esquina superior derecha y se desplaza cada vez una neurona a la derecha y al terminar se repite el proceso ahora empezando en el segundo renglon de neuronas y así sucesivamente hasta cubrir toda la imagen, la capa convolucional deberá contar con un arreglo de 24x24 neuronas. Sin embargo el tamaño de la capa convolucional no tiene por qué ser forzosamente este, puede cambiar al aplicar dos modificaciones que se conocen como padding y stride. Stride define cuantas unidades se tiene que desplazar el kernel en cada paso y padding le permite sobrepasar las fronteras del arreglo de la capa anterior llenando las entradas inexistentes con valores que pueden estar definidos a partir de diferentes criterios, por ejemplo el valor del pixel más

cercano o un promedio de los N pixeles más cercanos. Hasta ahora se trató la estructura de las capas convolucionales pero aún no se menciona como las neuronas procesan la informacion que reciben. Para explicar esto se sigue con el ejemplo de [15] donde la salida de la neurona j, k de la capa convolucional está dada por:

$$\sigma(b_{j,k} + \sum_{l=0}^{4} \sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}) \tag{3.5}$$

Donde σ es una función de activación, que para capas convolucionales es comúnmente la identidad. $b_{j,k}$ es el *bias* de la neurona correspondiente, los parámetros $w_{i,j}$ conforman el kernel y el elemento $a_{i,j}$ es la salida de la neurona i, j de la capa anterior.

De la expresion 3.5 se pueden hacer algunas observaciones:

- Se puede pensar a la capa convolucional como una sola fila de neuronas en lugar de un arreglo, con 2 diferencias respecto a la capa totalmente conectada que ya se describió. La primera es que una neurona no recibe información de todas las neuronas de la capa anterior y la segunda es que las neuronas comparten parámetros lo que de cierta forma es una ventaja ya que se reduce el número de operaciones que se tienen que realizar.

- Retomando la última parte de la observación anterior, es claro que todas las neuronas de la capa convolucional comparten el mismo kernel. Esto es una de las características que hacen especialmente buenas a las capas convolucionales en algunos campos como clasificación de imágenes y detección de objetos. Una capa convolucional puede contar con varios kernels y cada uno de estos se puede especializar en la detección de un cierto patrón lo que vuelve el modelo muy robusto para las tareas ya mencionadas entre otras.

Capa de reducción (pooling):

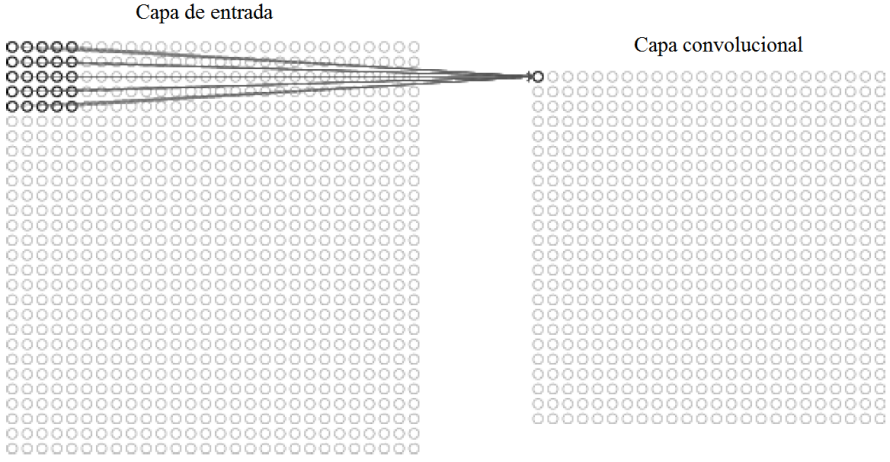


Figura 3.3: RNA convolucional [15]

Para entender el funcionamiento de este tipo de capa pensemos en la misma estructura de una capa convolucional. La única diferencia es que ahora la salida de la neurona j, k estará dada por:

$$y_{j,k} = \sigma(a_{j,k}^{\vec{}}) \tag{3.6}$$

Donde $a_{j,k}^{\vec{}}$ se utiliza para designar los elementos de la capa anterior que están dentro de la region del kernel y σ es una función de esos elementos. La función σ más común es la que toma el elemento más grande de $a_{j,k}^{\vec{}}$ y en este caso la capa se conoce como max-pooling.

3.2.4. Aprendizaje de RNAs

Hasta ahora se han descrito las RNAs de forma estructural y el proceso que usan para procesar los datos. Sin embargo falta explicar lo que es probablemente el punto más importante que es la forma en la que aprenden, que como ya se ha mencionado, se realiza a través del ajuste de los pesos sinápticos o parámetros. El tipo de aprendizaje de las RNAs se puede dividir en dos grandes categorías: aprendizaje supervisado y no supervisado [4].

En el primero, el entrenamiento se lleva a cabo en base a un conjunto de pares de datos de entrenamiento. Cada par de entrenamiento consta de dos conjuntos de datos. El primero son los que recibirá la RNAs como entrada mientras que el segundo conjunto son las salidas correctas que debería tener como resultado. En el segundo tipo de aprendizaje no se cuenta con estos datos de entrenamiento. Para este trabajo únicamente se utilizaron RNAs con aprendizaje supervisado y toda la descripción que se hace a continuación corresponde a esta categoría. El proceso de aprendizaje supervisado se puede dividir en dos pasos sucesivos. El primero se conoce como propagación hacia atrás o back propagation y el segundo es un algoritmo de optimización o simplemente un optimizador. Aunque puede ser anti intuitivo vale la pena empezar explicando el segundo paso ya que hace la comprensión del proceso de aprendizaje mucho más fácil.

Optimizadores

Como ya se mencionó, en el aprendizaje supervisado se cuenta con un conjunto de datos de entrenamiento que se usa para entrenar un modelo de RNA. Entonces dado un conjunto con N datos de entrenamiento $T = \{(\vec{x_i}, \vec{t_i})\}_{i=1}^{N}$ donde cada $\vec{x_i}$ representa un dato de entrada de la RNA y $\vec{t_i}$ la salida correcta correspondiente, se obtendrá otro conjunto $Y = \{\vec{y_i}\}$ que corresponde a las salidas reales de la RNA al procesar cada conjunto de datos $\{\vec{x_i}\}$. Lo que se hace es definir una función $E(W, b)$, que se conoce como función de perdida, y que depende los pesos sinápticos (W) y los *bias* (b). Una de las funciones de perdida que se usa comúnmente es el error cuadrático medio o ECM:

$$E(W, b) = \frac{1}{2} \sum_{i=1}^{N} (t_i - y_i)^2 \tag{3.7}$$

La función de perdida es siempre derivable y generalmente proporciona una forma de comparar que tan cercana es la respuesta de la red y_i a la respuesta correcta t_i. Claramente el objetivo es que la respuesta de la RNA sea lo más cercano posible a la correcta, y esto se traduce en minimizar la función de perdida cuando esta es convexa, que es el caso de 3.7 y de la mayor parte de funciones de perdida usadas. Lo que se haría normalmente para solucionar el problema es encontrar el mínimo global de la función E que para el caso

de 3.7 corresponde a $E = 0$. Siendo que las RNAs que se usan normalmente cuentan con un gran número de parámetros la búsqueda del mínimo puede ser computacionalmente muy costosa y en muchos casos prácticamente imposible. Sin embargo no todo está perdido. Primero hay que pensar que muchas veces no se necesita encontrar el mínimo global. Hay problemas que no necesitan soluciones exactas, sino una solución que permite un cierto margen de error y probablemente existan ciertos mínimos locales que cumplan con ese margen. Con esta idea en mente la pregunta es ahora ¿Cómo se puede encontrar un mínimo local sin calcular puntos de inflexión?, que también se convierte en un problema computacionalmente muy costoso, y la respuesta es: Con la información del gradiente de $E(W, b)$ que se denotara como ∇E. Para dejar clara está última idea se explica a continuación porqué la información de ∇E puede ayudar. Supóngase una función convexa $h(\vec{a})$ que depende de un cierto número de variables $\vec{a} = (a_1, a_2, ...)$. Si se calcula el gradiente de h en un punto $\vec{a}^{(0)}$, es decir $\nabla h(\vec{a}^{(0)})$, este último apuntará en la dirección de mayor cambio de h. Esto quiere decir que si se escoge un nuevo punto $\vec{a}^1 = \vec{a}^0 + \Delta\vec{a}^0$ con $\Delta\vec{a}^{(0)}$ pequeño y que apunte en el sentido negativo de la dirección de $\nabla h(a^{(0)})$ se tendrá $h(\vec{a}^{(1)}) < h(\vec{a}^{(0)})$. Siguiendo este procedimiento de forma iterativa se puede esperar que para alguna t el punto $\vec{a^{(t)}}$ este muy cerca de algún mínimo. Justo este procedimiento iterativo que va tomando puntos en el sentido negativo del gradiente de la función de error es el que realizan los optimizadores. El desplazamiento en cada paso se puede hacer de diferentes maneras y por esta razón existen diferentes optimizadores. Uno de los más famosos recibe el nombre de descenso del gradiente y sigue la siguiente regla [1]:

$$W^{(t+1)} = W^{(t)} - \eta \nabla E(W^{(t)}) \tag{3.8}$$

El parámetro $\eta > 0$ se conoce como tamaño de paso y se encarga de modular que tanto se desplaza el punto en cada iteración. Se tiene que tomar en cuenta que la función de perdida E está definida a partir de todo el conjunto de entrenamiento y por lo tanto para dar un paso se tienen que procesar todos los datos con la RNA. Este método no siempre es el mejor, en especial cuando los datos de entrenamiento tienen cierta información redundante, lo cual sucede con frecuencia. Este problema se puede solucionar usando un optimizador que es una pequeña variante del descenso del gradiente y se conoce como descenso

del gradiente estocástico. Lo único que cambia es que para cada actualización de los pesos sinápticos se usa un subconjunto de los datos de entrenamiento en lugar de todos. De esta forma se define una nueva función de perdida:

$$E_C(W, b) = \frac{1}{2} \sum_{i \epsilon C} (t_i - y_i)^2 \tag{3.9}$$

Donde C es un subconjunto de T. Lo que se hace normalmente es definir la dimensión para C y dividir T en tantos conjuntos excluyentes de esta dimensión se pueda e ir realizando actualizaciones de los pesos sinápticos con cada uno de estos conjuntos.

Ahora hay que hacer un poco de énfasis en el tamaño de paso. Este tiene gran importancia ya qué va a determinar en gran parte que tan rápido y hasta qué punto puede aprender una cierta RNA. Y esto no es poca cosa, la diferencia en el entrenamiento de la misma RNA con dos tamaños de paso diferentes puede ser abismal. Es por esto que gran parte del desarrollo del campo se ha centrado en este tema. Tan es así que en los últimos años ha surgido lo que ahora se conoce como optimizadores de tamaño de paso adaptables. Lo que han logrado estos optimizadores es, como su nombre lo indica, modificar el tamaño de paso en cada actualización de los pesos sinápticos para acelerar el proceso de aprendizaje. Hay algunos de estos optimizadores que incluso han logrado adaptar el tamaño de paso para cada peso sináptico. Algunos de estos nuevos optimizadores son AdaGrad, RMSProp y Adam cuyo funcionamiento se puede encontrar en [6].

Para terminar esta sección vale la pena comentar solo un idea más. En la práctica cuando se realiza el aprendizaje es común usar varias veces cada dato de entrenamiento ya que con una sola vez puede no obtenerse el resultado esperado. Lo que se hace entonces es dividir el entrenamiento en épocas. Durante una época se ocupan todos los datos de entrenamiento sin repetir ninguno. Una vez se han usado todos se dice que se terminó la época de entrenamiento y se avanza a la siguiente y así sucesivamente hasta que se obtiene el resultado esperado. También es común acomodar de forma aleatoria los datos en cada época.

Propagación hacia atrás

Hasta ahora se ha descrito el segundo paso del proceso de aprendizaje, pero se ha dejado un gran vacío por llenar: ¿Cómo calcular de forma eficiente el valor de ∇E en un punto en el espacio de parámetros?. Es justamente esto lo que resuelve el algoritmo de back propagation. A muy grandes rasgos lo que hace este algoritmo es ir calculando las derivadas parciales de la función de perdida con respecto a cada peso sináptico. Esto lo hace empezando por las contribuciones de la última capa, siguiendo a la penúltima y así sucesivamente hasta llegar a la capa de entrada. Este proceso es lo que le da el nombre de propagación hacia atrás. No se profundizará más en la descripción del algoritmo pero si se quiere tener una descripción más formal de este se puede consultar [1].

3.3. Redes Neuronales Artificiales y sensado comprimido

En el capítulo anterior se explica en que consiste el sensado comprimido en términos generales y el problema de minimización que se intenta solucionar el cual se encuentra resumido en la expresión 2.8. Comúnmente para resolver este problema se utilizan diferentes algoritmos iterativos, pero en años recientes surgió un nuevo enfoque. Este consiste en aplicar redes neuronales artificiales para atacar el problema y los resultados han sido buenos. Lo primero que hay que mencionar es que cuando se usan redes neuronales se pierde el control en una parte del algoritmo. Para explicar esto, primero hay que recordar la expresión 2.8, a partir de la cual se puede explicar el proceso que se sigue para resolver el problema usando RNAs. Lo primero que se hace es construir la matriz de muestreo, es decir Φ_m. Una vez la matriz construida, el siguiente paso es muestrear la señal. Es hasta este último paso donde los métodos son iguales. En el caso iterativo lo que sigue es resolver 2.8 con algún algoritmo de minimización, para lo que se necesita conocimiento de la matriz de representación Ψ. Pero en el caso de la RNA lo único que se hace es entregarle el muestreo, esperar a que lo procese y recibir el resultado final. Lógicamente la RNA utilizada tuvo un entrenamiento previo que le permite generar un resultado razonable, este proceso se explica más adelante. Esta última parte que representa en si la reconstrucción de la señal, es donde se pierde el con-

trol en el caso de la RNA. Una vez entrenada, no es posible saber cómo está manipulando el muestreo de la señal y por esta misma razón no hay forma de modificarla para obtener mejores resultados. Esta es una desventaja que tienen los modelos basados en RNAs, que hasta ahora no se puede superar.

Capítulo 4

Modelos

En este capítulo se realiza una descripción detallada de los modelos que se utilizan para realizar la reconstrucción de señales que implica el método de sensado comprimido, a partir del muestreo de estas. Estos modelos están compuestos por RNAs y conllevan un proceso de entrenamiento con datos. Este proceso así como el tipo de datos utilizados también se explican. Finalmente se ahonda un poco más en dos parámetros importantes que tienen un fuerte impacto en el desempeño general y la forma en que estos fueron seleccionados.

4.1. Estructura de los modelos

En esta tesis se prueban 5 estructuras diferentes de RNA para realizar la reconstrucción de imágenes digitales, que son el tipo de señales con las que se trabaja. Tres de estas se obtienen directamente de [14], [11] y [20] y se hace referencia a ellas como StanNet, ReconNet y DR2Net correspondientemente. La cuarta, nombrada LinearNet, corresponde a una capa que se utiliza en ReconNet y DR2Net, mientras que la quinta, nombrada ReconNet2, tiene la misma estructura que ReconNet pero su primera capa es preentrenada igual que se hace en DR2Net. Antes de pasar a la descripción de cada una de las estructuras de las redes, se muestra la notación usada para hacer referencia a los diferentes tipos de capas utilizadas:

- Conv2D(m,k): Capa convolucional con m filtros de tamaño kxk que

extiende, con ceros, el tamaño del tensor de entrada de tal forma que la salida conserve el tamaño original.

- Densa(n): Capa totalmente conectada con n neuronas.
- Plana: Capa que recibe un tensor y lo reduce a una dimensión concatenando las entradas de la última dimensión a la primera.
- Activación(f): Capa que recibe un tensor $\vec{x}$ y regresa otro del mismo tamaño pero a cada entrada le aplica la función f.
- MaxPool2D(n): Capa que realiza la operación de MaxPooling con un kernel de nxn a las primeras 2 dimensiones de un tensor, extendiendo, con ceros, su tamaño de tal forma que el resultado conserve el tamaño original.
- BatchNorm: Recibe un tensor y normaliza sus entradas.
- Reshape(dims): recibe un tensor y cambia sus dimensiones a dims siempre y cuando sea compatible con el tensor recibido. En este trabajo se utiliza más que nada para cambiar un tensor de una dimensión de tamaño 1024 a un tensor de 2 dimensiones de tamaño 32x32.

Ahora que se tiene esta notación se puede pasar a la descripción de la arquitectura de las diferentes RNAs que se utilizan a lo largo de esta tesis.

StanNet

La primera parte de esta red consta de dos bloques consecutivos con la siguiente estructura: Primero cuentan con un par de capas convolucionales con 16 filtros cada una de dimensiones 5x5. Posteriormente se normalizan los tensores resultantes y se aplica a ellos la función "ReLu" al final del bloque se realiza una reducción de dimensiones con un filtro de 2x2. De forma concreta la estructura de uno de estos bloques se puede resumir como: Conv2D(16,5) - Conv2D(16,5) - BatchNorm() - Activation("relu") - MaxPool2D(2). Después de los dos bloques hay una capa Plana y finalmente como capa de salida una totalmente conectada con 1024 neuronas (Densa(1024)).

ReconNet y ReconNet2

Cuenta primero con una capa totalmente conectada de 1024 neuronas (Densa(1024)). El vector resultante de esta primera capa se redimensiona a una matriz de 32x32 entradas. Posteriormente cuenta con dos bloques compuestos por 3 capas convolucionales donde al resultado de cada una de estas se le aplica la función "ReLu", excepto a la última capa del segundo bloque. La primera capa convolucional de cada bloque tiene 64 filtros de tamaño 11x11, la segunda 32 filtros de 1x1 y la última un filtro de 7x7. La estructura de un bloque se puede resumir de la siguiente manera: Conv2D(64,11) - Activación("relu") - Conv2D(32,1) - Activación("relu") - Conv2D(1,7) - Activación("relu"). La última capa de este modelo lo único que hace es convertir la matriz de 32x32 que recibe en un vector de 1024 entradas.

LinearNet

Esta red es la más sencilla de todas y cuenta únicamente con una capa totalmente conectada de 1024 neuronas y al vector que resulta de esta capa lo convierte en una matriz de 32x32.

DR2Net

Esta red tiene primero una capa totalmente conectada de 1024 neuronas y al vector resultante lo convierte en una matriz de 32x32. Posteriormente cuenta con 4 bloques que tienen la siguiente estructura: Una capa convolucional de 64 filtros de 11x11 seguida de la aplicación de la función "ReLu" y posterior normalización del resultado. Se repite este proceso pero ahora la capa convolucional tiene 32 filtros de 1x1. Por último se tiene nuevamente este proceso, sin la normalización, con 1 filtro de 7x7 para la capa convolucional. La estructura de un bloque se resume de la siguiente manera: Conv2D(64,11) - Activación("relu") - BatchNorm()- Conv2D(32,1) - Activación("relu") - BatchNorm- Conv2D(1,7) - Activación("relu"). En su parte final la red convierte la matriz de 32x32 en un vector de dimensión 1024. La particularidad de este modelo es que los tensores no fluyen de forma consecutiva siempre. El segundo bloque recibe como entrada la suma, entrada a entrada, de la entrada y salida del primer bloque. El tercer bloque recibe como entrada la suma, entrada a entrada, de la entrada y salida del segundo bloque. El 4 bloque recibe como

entrada la suma, entrada a entrada, de la entrada y salida del tercer bloque. Y finalmente la capa Plana() recibe como entrada la suma, entrada a entrada, de la entrada y salida del bloque 4. En la figura 4.1, la cual fue tomada de [20], se puede observar una representación gráfica del modelo. Los autores de esta arquitectura argumentan que si la primera capa realiza una reconstrucción preliminar de la señal, y teniendo la estructura no lineal antes descrita, los bloques de capas convolucionales aprenden una especie de residuo entre la reconstrucción preliminar y la señal verdadera. Según ellos para la RNA esto es más fácil de aprender y por esta razón puede llegar a tener un mejor desempeño.

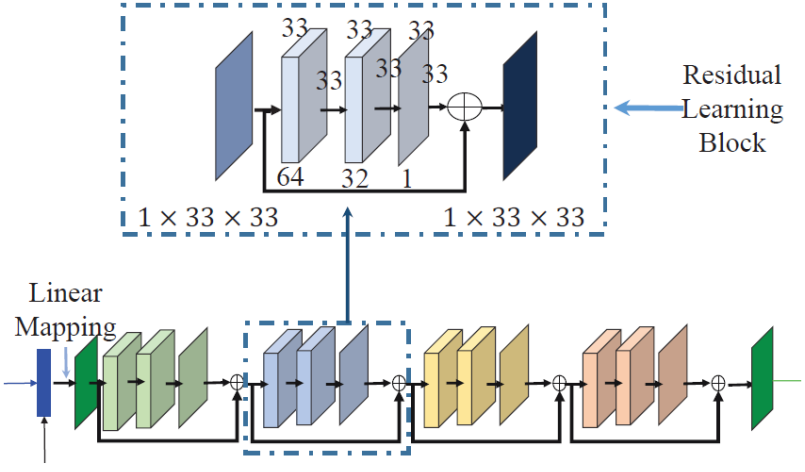


Figura 4.1: DR2Net

Por último hace falta mencionar el estado inicial de las redes, es decir que valores tienen sus parámetros cuando se inicia el entrenamiento. Por defecto el paquete de programación que se utiliza, el cual se especifica más adelante, inicializa los parámetros con una distribución normal centrada en 0 y varianza pequeña lo cual no se modifica. Las únicas excepciones son las redes ReconNet2 y DR2Net a las cuales se les precargan los pesos de su primera capa con los de la red LinearNet correspondiente para cada porcentaje. De esta forma se tiene que la primera capa de estas estructuras produce realmente una reconstrucción

preliminar de la señal.

4.2. Entrenamiento de los modelos

Cuando se define una RNA algo que se fija es la dimensión del tensor de entrada que en este caso tiene la información del muestreo. Esta dimensión depende y define el porcentaje de muestreo que puede procesar la RNA. Y por porcentaje de muestreo se hace referencia al número de elementos que tiene el muestreo de una señal. Si se está trabajando con imágenes de 1024 pixeles, un muestreo del 10 %, por ejemplo, corresponde a uno que obtiene 102 (aproximadamente el 10 % de 1024) mediciones de la imagen. Para poder observar el desempeño del sensado comprimido y de los diferentes modelos, en esta tesis se trabaja con 5 diferentes porcentajes de muestreo que son los siguientes: 1 %, 5 %, 10 %, 20 % y 40 %. Esto significa que para cada estructura de RNA se tendrán en realidad 5 modelos.

El proceso de entrenamiento para todas las estructuras es equivalente. Cada RNA se corre por 1000 épocas utilizando ECM como función de perdida (expresión 3.7) y corresponde a un entrenamiento supervisado, lo que significa que se lleva a cabo a partir de ejemplos con los que la RNA logra "aprender" la tarea que tiene que realizar, en este caso la reconstrucción de una imagen. Los ejemplos con los que se entrena cada RNA están conformados por un conjunto $\{(\vec{y_k'}, \vec{f_k})\}_{k=1}^{N}$ de pares de datos. Los elementos $\vec{f_k}$ son señales completas, que en el caso de este trabajo corresponden a imágenes digitales de 1024 pixeles (32x32). En la figura 4.2 se pueden observar algunas de las imágenes que se usan para el entrenamiento. Por otro lado los elementos $\vec{y_k'}$ son las entradas que reciben los modelos y guardan la información del muestreo de la señal. Para las redes ReconNet, ReconNet2, DR2net y LinearNet $\vec{y_k'}$ corresponde directamente al muestreo. Sin embargo para StanNet, respetando la implementación original, esta configuración cambia. Para esta red $\vec{y_k'}$ se construye aplicando la matriz $T = \Phi_m^T$ (transpuesta de la matriz de muestreo) al muestreo y posteriormente un Reshape((32,32)). De esta forma para StanNet la entrada no es un vector sino una matriz que tiene las mismas dimensiones que la señal original que es una imagen de 32x32. La ventaja que ven los autores en esta arquitectura es que no se tiene que modificar cuando se trabaja con diferentes porcentajes de muestreo.

Hay que mencionar que en los artículos de los cuales se obtienen las estructuras de las redes, la matriz de muestreo Φ_m es una matriz aleatoria construida a partir de una distribución normal con alguna varianza. En esta tesis la matriz de muestreo también es aleatoria pero para construirla se utiliza una distribución Bernoulli simétrica con elementos 1 y -1. Otra forma de lograr esto sería usando una distribución normal centrada en 0 y tomar todos los valores positivos como 1 y los negativos como -1. Se utiliza este tipo de distribución en específico debido a que a la hora de utilizar la matriz de muestreo en el arreglo experimental, el cual se describe en el siguiente capítulo, únicamente se pueden representar dos valores diferentes.

Figura 4.2: Ejemplos de imágenes digitales pertenecientes al conjunto CIFAR [10]

El número de pares del conjunto de entrenamiento varía dependiendo la estructura de RNA usada. Para el caso de LinearNet es de 50,000 y para los demás modelos es de 10,000. Las imágenes se obtienen del conjunto CIFAR [10], que está conformado por imágenes con una dimensión de 32x32 pixeles. Otro subconjunto de 10,000 imágenes diferentes, también de CIFAR y al cual se hace referencia como conjunto CIFARtest, se utiliza como validación a la hora del entrenamiento de las RNAs. Definiendo la dimensión de las imágenes con las que se realiza el entrenamiento define automáticamente la dimensión de las imágenes que los modelos son capaces de reconstruir. Esta es la razón por la cual en esta tesis se trabaja siempre con una dimensión de 32x32 o dicho de otra forma imágenes de 1024 pixeles.

La implementación de los modelos se realiza en el lenguaje de programación Python utilizando principalmente la librería de Keras con Tensorflow como backend y con la ayuda de una tarjeta de procesamiento gráfico NVIDIA Tesla K80. Más adelante se tiene un ejemplo simplificado del código principal. Cada que se entrena un modelo se guarda un documento csv con el valor de la función de perdida para el conjunto de entrenamiento y validación por cada época además del tiempo que tardo en correr cada una de estas. También se guarda el estado del modelo cada 20 épocas para poder reutilizarlo y no tener que volverlo a entrenar. En otros casos, donde la función de perdida no disminuye de forma suave se guardó el estado de la RNA únicamente donde el error para el conjunto de validación fuera menor con respecto a estados pasados. Esto se aplicó sobre todo para el modelo DR2Net.

Como se comenta en la sección 3.2.4, para realizar el entrenamiento supervisado de una RNA se emplea un optimizador, el cual se encarga de actualizar sus parámetros. Para este trabajo se utilizó el optimizador Adam. Este cuenta con varios parámetros con los que se puede modificar su funcionamiento, sin embargo en esta tesis únicamente se modifican 2: El tamaño de paso y su decaimiento. Este último se encarga de disminuir el valor del tamaño de paso a lo largo del entrenamiento.

4.2.1. Tamaño de paso

El tamaño de paso es un parámetro sumamente importante. Un valor muy grande puede hacer la convergencia del modelo prácticamente imposible mientras que en el otro extremo, valores muy cercanos a cero la pueden hacer extremadamente lenta. Para realizar la búsqueda del mejor candidato a tamaño de paso de cada modelo, el decaimiento se fija en cero y posteriormente se entrena cada uno por 100 épocas 3 veces, cada una de estas cambiando el tamaño de paso en factores de 10, empezando en 10^{-3} y terminando en 10^{-5}. En la figura 4.3 se tiene un ejemplo de este procedimiento para el modelo LinearNet para el caso de 10 %, en esta ocasión se agregaron dos valores más: 10^{-2} y 10^{-6} para ver de forma más clara el efecto del tamaño de paso. Como se observa, este parámetro es sumamente importante ya que impacta directamente en el desempeño del modelo lo cual se ve reflejado en el valor de

la función de perdida (ECM) conforme las épocas de entrenamiento avanzan. Este parámetro puede ser la diferencia entre tener un entrenamiento eficiente a uno que puede tomar demasiado tiempo o incluso jamás converger a un estado deseable.

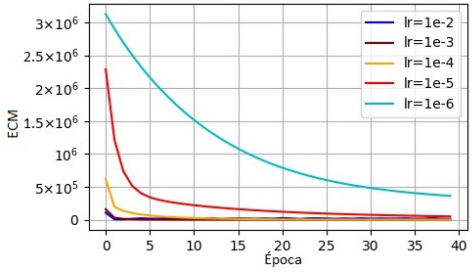


Figura 4.3: Evolución de la función de perdida (ECM) del modelo LinearNet para el caso 10 % para diferentes tamaños de paso y sin decaimiento.

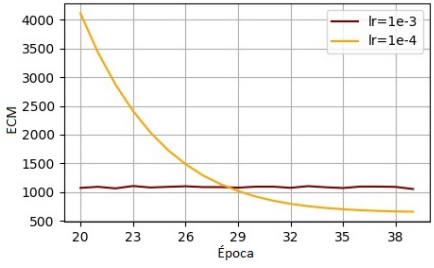


Figura 4.4: Evolución de la función de perdida (ECM) del modelo LinearNet para el caso 10 % para los dos mejores tamaños de paso y sin decaimiento.

En la figura 4.4 se muestra el entrenamiento para los dos mejores tamaños de paso para el modelo LinearNet para el caso de 10 % que resultaron ser: 10^{-3} y 10^{-4}, siendo el último mejor. Se podría hacer una búsqueda más fina de un tamaño de paso más eficiente, sin embargo esto representaría mayor tiempo de cómputo por una mejor eficiencia si, pero quizá no significativa. Además usando el parámetro de decaimiento se puede encontrar una solución parecida e incluso mejor en ciertas ocasiones y sin la desventaja del aumento en tiempo de cómputo.

4.2.2. Decaimiento del tamaño de paso:

El decaimiento permite a una RNA, en ciertas ocasiones, un proceso de entrenamiento más eficiente e incluso con mejores resultados. Su objetivo es disminuir, en algún factor, el tamaño de paso cada vez que se actualizan los pesos del modelo, de esta forma cuando este se encuentra cerca de algún mínimo puede realizar actualizaciones mucho más finas para acercarse a él. En el caso particular del optimizador Adam y el módulo de Keras en Python este decaimiento modifica el tamaño de paso de la siguiente forma:

$$\eta = \frac{\eta_0}{1 + d \cdot i} \tag{4.1}$$

Donde η_0 es el tamaño de paso inicial, d es el valor del decaimiento, i es un contador del número de veces que se han actualizado los parámetros de la RNA y η es el tamaño de paso real.

Lo que se busca con el decaimiento es permitir una búsqueda más fina cuando el modelo se encuentra cerca de un mínimo, o estancado en alguna región. Sin embargo si este decaimiento es demasiado grande puede volver el movimiento del estado del modelo muy lento y aumentar el tiempo de entrenamiento considerablemente lo cual es claramente indeseable. La forma en la que se selecciona el decaimiento en este caso parte del entrenamiento previo donde no se usa decaimiento. Se toma el mejor tamaño de paso y se observa alrededor de qué época se empieza a estancar la curva de la función de error y con esta información se puede seleccionar un valor para el decaimiento. Este proceso no es exacto, por lo que a veces la selección no es la correcta y el valor del decaimiento tiene que ser modificado.

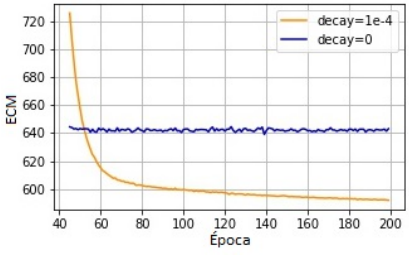


Figura 4.5: Influencia del decaimiento del tamaño de paso. Se muestra la evolución de la función de perdida (ECM) para el modelo LinearNet para el caso 10 % para dos configuraciones, una con decaimiento y la otra sin este.

Para ilustrar el impacto que puede tener el decaimiento en el proceso de entrenamiento se puede observar la figura 4.5. En ella se tiene el entrenamiento del modelo LinearNet para el caso 10 % con dos configuraciones diferentes para el optimizador Adam. Ambas tiene el mismo tamaño de paso, pero el primero no tiene decaimiento mientras que el segundo si. Es claro, desde la

época 60, que el caso sin decaimiento está estancado en un valor de alrededor de 640 del cual no se mueve en 140 épocas. Por otra parte cuando se agrega decaimiento el modelo sigue mejorando por estas mismas 140 épocas y al final continúa con una tendencia a la baja, lo que significa probablemente, que puede seguir mejorando. En este caso en particular, en 200 épocas, el decaimiento logra bajar el error de entrenamiento del modelo cerca de un 8 % comparado al caso sin decaimiento.

A continuación se muestra una tabla donde vienen especificados los valores finales de tamaño de paso (TP) y decaimiento (D) que se usan para cada modelo.

	Stanford	ReconNet	LinearNet	ReconNet2	DR2Net
1 %	TP: 10^{-3} D: 0	TP: 10^{-3} D: 0	TP: 10^{-3} D: 10^{-4}	TP: 10^{-3} D: 10^{-3}	TP: 10^{-3} D: 10^{-4}
5 %	TP: 10^{-3} D: 0	TP: 10^{-4} D: 10^{-5}	TP: 10^{-4} D: 10^{-4}	TP: 10^{-3} D: 10^{-4}	TP: 10^{-3} D: 10^{-3}
10 %	TP: 10^{-3} D: 0	TP: 10^{-4} D: 10^{-4}	TP: 10^{-3} D: 10^{-3}	TP: 10^{-3} D: 10^{-3}	TP: 10^{-4} D: 10^{-4}
20 %	TP: 10^{-3} D: 0	TP: 10^{-4} D: 0	TP: 10^{-3} D: 10^{-3}	TP: 10^{-3} D: 10^{-4}	TP: 10^{-4} D: 10^{-3}
40 %	TP: 10^{-3} D: 0	TP: 10^{-4} D: 0	TP: 10^{-3} D: 10^{-3}	TP: 10^{-3} D: 10^{-4}	TP: 10^{-4} D: 10^{-4}

Cuadro 4.1: Parámetros del optimizador Adam para el entrenamiento de los diferentes modelos

4.3. Ejemplo en código

Las descripciones de la estructura de los modelos y su entrenamiento dejan muchas cosas claras, sin embargo muchas veces es bueno tener una idea de cómo se traduce esto a código. Esto permite, algunas veces, entender más claramente cómo se obtuvieron los resultados y aterrizar un poco más ideas que quizá no se entendieron muy bien.

En la figura 4.6 podemos ver una versión simplificada pero que ilustra muy bien cómo se construye el modelo LinearNet y posteriormente se entrena. En las líneas 1-3 se cargan los modulos y funciones que se usan, en los que se pueden destacar Dense y Reshape que representan las capas que aquí se han

```python
# importar modulos que se usarán
from keras.models import Sequential
from keras.layers import Dense, Reshape
import numpy as np

# cargar muestreos (10%) de imágenes de entrenamiento y validación
y_train = np.load('y_train.npy')
y_val   = np.load('y_val.npy')

# cargar imágenes digitales de entrenamiento y validación (CIFAR)
f_train = np.load('f_train.npy')
f_val   = np.load('f_val.npy')

# Damos a los muestreos e imágenes las dimensiones requeridas
y_train = y_train.reshape(50000,10)
y_val   = y_val.reshape(50000,10)
f_train = f_train.reshape(50000,1024)
f_val   = f_val.reshape(50000,1024)

# crear modelo
modelo = Sequential()

# agregar capas del modelo
modelo.add(Dense(1024, input_dim=10))
modelo.add(Reshape((32,32,1)))

# compilar modelo con el optimizador y función de error deseados
modelo.compile(optimizer='adam', loss='mse')

# entrenar el modelo
modelo.fit(y_train, f_train, validation_data=(y_val, f_val), epochs=1000)
```

Figura 4.6: Ejemplo de implementación simple y entrenamiento de modelo LinearNet en Python

denominado como Densa y Reshape respectivamente. De las líneas 7-12 se cargan los muestreos, en este caso correspondientes a 1 %, y sus imágenes correspondientes. Hay que transformar muestreos e imágenes a dimensiones compatibles con el modelo lo cual se realiza en las líneas 15-18. Lo siguiente es definir el modelo a partir de la clase Sequential. En las líneas 24 y 25 se agregan las capas correspondientes a LinearNet, especificando en la primera el tamaño del vector de entrada que en este caso es 10 (muestreo de 1 %). El penúltimo paso es compilar el modelo (línea 28) en donde se especifica el optimizador y la función de perdida y finalmente en la línea 31 se entrena el modelo, en este caso por 1000 épocas.

Capítulo 5

Resultados

En este capítulo se encuentran los resultados más importantes de esta tesis. Por una parte se describe el arreglo experimental que se utiliza así como sus componentes. También se discute su funcionamiento, el tipo de datos que se obtienen con él, los cuales tienen la información del muestreo, y como se pueden utilizar estos para realizar la reconstrucción final de la imagen deseada. Por otra parte se presentan los resultados obtenidos que no son más que reconstrucciones de algunas imágenes seleccionadas. Se prueban 3 técnicas de reconstrucción diferentes que como se verá más adelante llevan a resultados muy distintos.

5.1. Arreglo experimental

Descripción y funcionamiento

Lo que se intenta es diseñar un arreglo experimental que permita dar una demostración sobre el proceso de muestreo de una señal usando el sensado comprimido. En este caso la señal corresponde a una imagen. A continuación se hace una descripción del montaje experimental, que es parecido al usado en [11] y que se puede observar en la figura 5.1, después se explica su funcionamiento.

El arreglo comienza con una fuente de luz Thorlabs M470L2 de longitud de onda de 470 nm conectada a un controlador de corriente para poder modi-

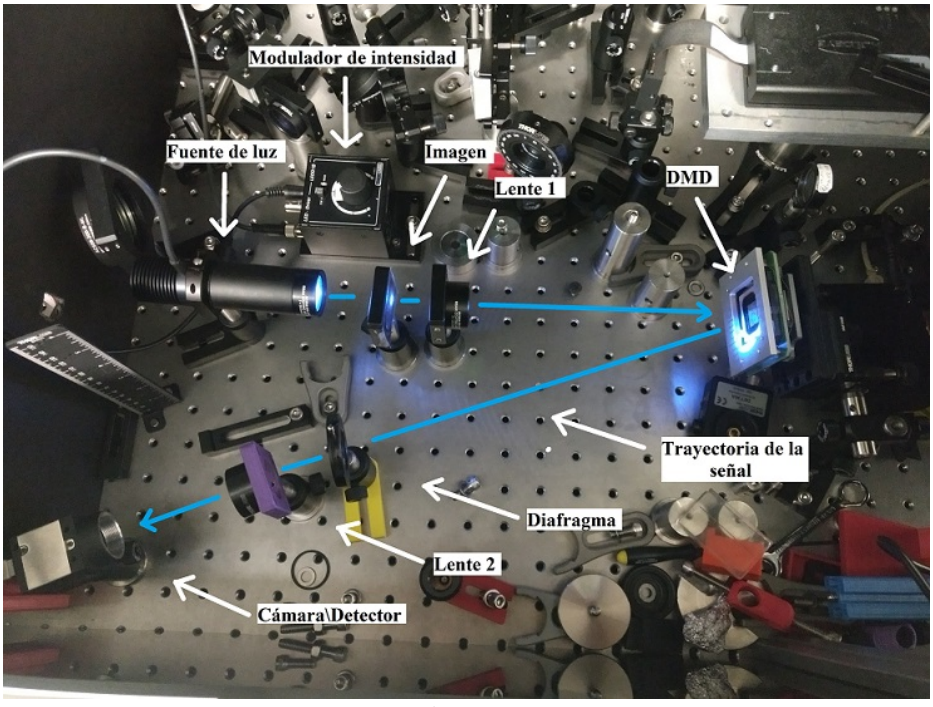
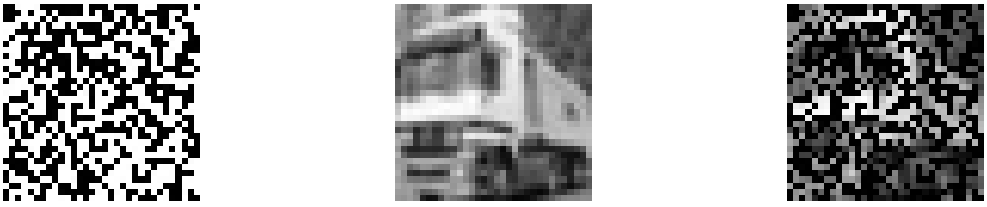


Figura 5.1: Arreglo experimetal

ficar su intensidad. La luz emitida por esta fuente llega a un acetato que tiene impresa con tinta negra alguna imagen. De esta forma las partes con tinta absorben la luz incidente mientras que en las otras regiones esta logra pasar formando así la representación de la imagen en la luz transmitida. Posteriormente una lente (lente 1) Thorlabs AC254-035-A-ML de distancia focal 35.0 mm permite enfocar la imagen transmitida en un dispositivo digital de microespejos ó DMD integrado en un sistema de control High-speed V-Modules de ViALUX. La superficie de este está formada por un arreglo de 1024x768 microespejos independientes que ocupan un área de 14.0x10.5 mm^2 y que funciona en un rango de espectro de 350nm a 2500nm. Cada uno de estos microespejos puede estar en dos posiciones diferentes con un ángulo relativo de 24 grados entre las dos. Para simplificar la explicación se dirá que los microespejos tienen dos estados ON y OFF. Si uno de ellos se encuentra en el estado



(a) (b) (c)

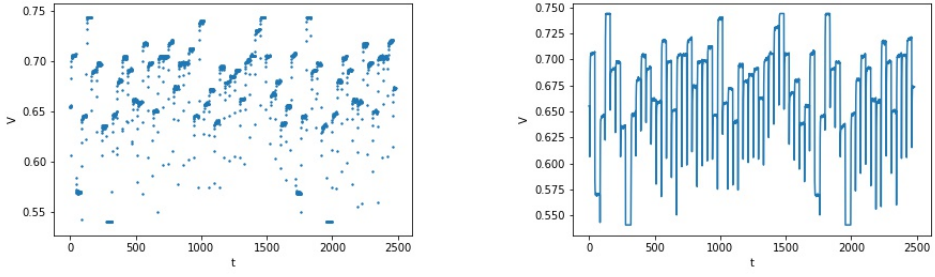
Figura 5.2: Funcionamiento de DMD

ON la luz que refleje llegará a un detector de intensidad de luz, mientras que si se encuentra en el estado OFF la luz saldrá en alguna otra dirección, no es importante cual solo que no llega al detector. Entre el detector y el DMD se coloca un diafragma y una segunda lente (lente 2) Thorlabs AC254-075-A-ML de distancia focal 75.0 mm que vuelve la imagen transmitida lo más cercano a un punto en la superficie del detector. El detector Thorlabs DET36A funciona en un rango de longitud de onda de 350-1100 nm y tiene un tiempo de subida del orden de 700 μs. Este va conectado a un osciloscopio Tektronix TDS2024C de tiempo de subida de 2.5 ns. Como última precisión hay que mencionar que el arreglo se puede modificar para usar una cámara Thorlabs DCC1645C en vez del detector y el osciloscopio. La única modificación que se tiene que hacer en el arreglo, además de la sustitución del detector por la cámara, es ajustar la distancia de la lente 2 para que se forme una imagen nítida en la cámara.

Lo que sigue es describir el funcionamiento del arreglo. La primera parte es muy clara y consiste en formar una imagen en la superficie del DMD, esta es la imagen que se intenta obtener. El siguiente paso es realizar el muestreo, para lo cual se realiza la suma de los elementos del producto, entrada a entrada, de la imagen en el DMD y cada elemento (renglón) de la matriz de muestreo. Esto está explicado en el capítulo 3 y resumido en la figura 2.1. Para este trabajo se toman dos representaciones de la imagen, una de tamaño 32x32 pixeles y otra de 64x64 que equivale a 4 bloques de 32x32. En el primer caso se utiliza un área total del DMD correspondiente a 736x736 microespejos donde cada pixel está representado por 23x23 de estos. En el segundo caso el área utilizada fue de 704x704 microespejos y cada pixel corresponde a 11x11 microespejos. Para poder realizar el muestreo físicamente lo que se hace es

cargar un renglón (equivalente a una imagen 32x32 pixeles) de la matriz de muestreo al área utilizada del DMD y poner un estado ON en los microespejos correspondientes a un pixel cuando a este le corresponda un 1 y OFF cuando le corresponda un -1. Esta es la razón por la cual se utiliza una distribución Bernoulli para generar las matrices de muestreo, genera únicamente dos elementos que se pueden representar con los dos estados del DMD. Si se hubiera usado una distribución normal para esto se habría tenido un número muy grande de elementos diferentes. Regresando al funcionamiento del arreglo, para hacer más claro el proceso de cargar un elemento de la matriz de muestreo al DMD se puede ver la figura 5.2. Se tiene un renglón (representado como una matriz de 32x32) de la matriz de muestreo en la figura 5.2a y un bloque de la imagen proyectada en el DMD en la figura 5.2b. Al realizar el procedimiento antes descrito, que consiste en superponer el renglón en la imagen, lo que se tiene en el DMD es la imagen que aparece en la figura 5.2c. La parte final del arreglo, que corresponde a la luz incidente al detector, realiza la última etapa del muestreo que suma la aportación de las 32x32 entradas. Finalmente se puede medir esto como un voltaje en el osciloscopio. Este proceso se tiene que realizar para cada renglón de la matriz de muestreo. Hay que mencionar que a cada una de estas se le agrega un elemento extra que corresponde a un renglón con un 1 en todas sus entradas. Este elemento da como resultado una medición de la intensidad total de la imagen.

Hasta ahora se tiene la descripción del arreglo y cómo funciona, sin embargo falta un pequeño detalle. Este concierne al DMD que tiene una funcionalidad que permite realizar el muestreo de una forma más sencilla. El controlador integrado al DMD permite cargar N diferentes arreglos de estados para los microespejos y escoger una frecuencia para correrlos en un bucle. De esta forma se pueden cargar, por ejemplo, 10 arreglos y correrlos a una frecuencia de 10 Hz y de esta forma en un segundo habrán pasado los 10. Lo interesante es que esto se puede observar y guardar como una señal en el osciloscopio para usarla posteriormente. Además a cada conjunto de arreglos que se carga al DMD se le añade uno extra que tiene cero en todas sus entradas; este se utiliza como referencia en la señal para saber cuándo empieza y termina la secuencia de interés.



(a) (b)

Figura 5.3: Porción de un muestro de la imagen 2 obtenido con el osciloscopio. Cada punto tiene como primera entrada el tiempo en el que fue adquirido y como segunda un voltaje que es proporcional a la intensidad de la señal que llega al detector. La distancia temporal entre puntos consecutivos es constante. En (a) se tiene los puntos como se obtuviero, y en (b) se unieron puntos consecutivos usando rectas

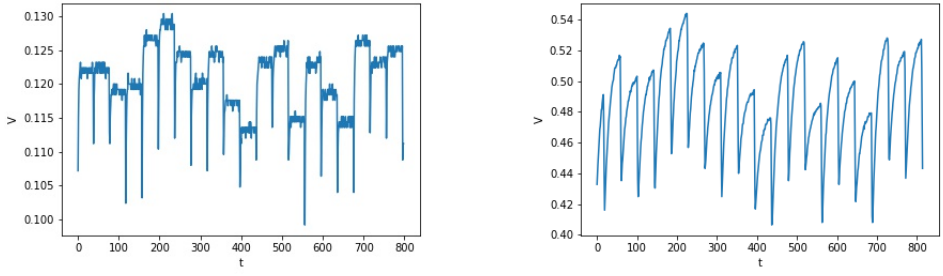
Descripción de los datos

Lo primero que se busca en esta sección es explicar en qué consisten los datos obtenidos con el arreglo experimental, los cuales se generan a partir de la señal proveniente de correr un conjunto de elementos de la matriz de muestreo en el DMD y reflejar la imagen en este. El osciloscopio, con el que se capta la señal en cuestión, entrega 2500 puntos por cada medición que se realiza. Estos puntos cuentan con dos coordenadas: Una temporal, que indica cuando se obtuvo ese dato y que aumenta en intervalos constantes, y otra que corresponde a la intensidad de la señal y se mide como un voltaje. En la figura 5.3a se puede observar una porción de una señal obtenida. En la figura 5.3b se tiene exactamente la misma señal pero ahora uniendo los puntos consecutivos con líneas. Esto permite ver más claramente la forma de las señales que se obtienen para los muestreos. Como se observa esta es una especie de señal de escalones separados por unos cuantos puntos. Los puntos pertenecientes a un escalón, que idealmente deberían ser iguales, representan el valor de un elemento del muestreo, de esta forma el conjunto de valores correspondientes a cada escalón conforman el muestreo en cuestión.

Si se recuerda la funcionalidad que se tiene para correr bucles de estados en el DMD a cierta frecuencia, se podría pensar que la mejor solución es co-

rrer a la frecuencia más alta para poder hacer el muestreo en el menor tiempo posible. Sin embargo hay que considerar 3 factores que podrían causar problemas en este escenario: El tiempo de subida del osciloscopio y del fotodetector que son 2.1 ns y del orden de 700 μs correspondientemente, y el tiempo de transición de estados del arreglo de microespejos el cual es cercano a 44 μs, lo que equivale a correr a la frecuencia máxima (22 727 Hz). Como se puede apreciar, el tiempo de subida del osciloscopio es varios órdenes de magnitud menor a los otros dos factores por lo que no hay preocupación de superarlo. Sin embargo donde sí se puede tener un problema es cuando la frecuencia utilizada en el DMD es muy grande ya que el tiempo que dura el arreglo de microespejos en un cierto estado puede ser menor al tiempo de subida o bajada del fotodetector y esto da como resultado una deformación de la señal. Hay que mencionar que en el manual del fotodetector no viene referencia al tiempo de bajada, lo cual sería únicamente relevante si este es mayor que el tiempo de subida. Para comprobar esto, y también observar la deformación que puede sufrir la señal, se cargan al microcontrolador del DMD algunos arreglos aleatorios, es irrelevante cuales sean, y se seleccionan dos frecuencias, 126 Hz y 1260Hz, para correrlos. Con la primera frecuencia cada estado del DMD tiene una duración de 7 936 μs, un orden de magnitud mayor al tiempo de subida del fotodetector. Por otro lado, con la segunda frecuencia el tiempo de cada estado es 793 μs lo cual es comparable al tiempo de subida del detector. Para cada una de estas frecuencias se obtiene la señal resultante con el osciloscopio.

En la figura 5.4 se puede observar una porción de la señal obtenida en cada caso, 126Hz y 1260Hz, que corresponde a 20 elementos. En estas dos figuras es claro cuál es el problema cuando se utilizan frecuencias muy altas. Lo que se busca obtener idealmente es una señal de escalones, donde cada uno de estos corresponde a un elemento del muestreo. Sin embargo cuando la frecuencia es muy alta estos escalones se deforman y se vuelve muy complicado asignar un valor fijo a cada uno. Quizá se podría hacer, pero se necesita desarrollar un método más complejo para hacerlo de forma automática y muy probablemente se tendría una gran incertidumbre sobre el valor obtenido. Además se puede observar que el tiempo de bajada es menor que el de subida que a final de cuentas es el que limita la frecuencia a la cual se puede correr el DMD. Por lo anterior la obtención de los muestreos se lleva a cabo con frecuencias no muy altas. Esto implica que para porcentajes grandes de muestreo, como 40 % que



(a) (b)

Figura 5.4: Datos obtenidos con el arreglo experimental para 20 elementos de la matriz de muestreo usando el arreglo únicamente con iluminación (sin imagen). En (a) se obtuvo del DMD corriendo a una frecuencia de 126Hz mientras que en (b) con una frecuencia de 1260Hz

cuenta con 409 elementos, el proceso no se pueda hacer en una sola medición y se tenga que dividir en secuencias donde se dividan el total de elementos en conjuntos más pequeños. El tamaño de estas secuencias y la frecuencia a la que se corren se presenta a continuación y varía dependiendo el porcentaje de muestreo.

- 1 %: 1 solo conjunto de 10 elementos. Corre a una frecuencia de 33Hz.
- 5 %: 1 solo conjunto de 51 elementos. Corre a una frecuencia de 156Hz
- 10 %: 1 conjuntos de 102 elementos. Corre a una frecuencia de 309Hz.
- 20 %: 2 conjuntos de 103 y 102 elementos correspondientemente. Corre a una frecuencia de 309Hz.
- 40 %: 3 conjuntos de 102 elementos y 1 de 103 elementos. Corre a una frecuencia de 309Hz.

A todas las secuencias se les agrega un arreglo de ceros al final para tomar como referencia del inicio y final de estas. Además se asegura que siempre aparezca este arreglo 2 veces en cada medición hecha con el osciloscopio.

Para poder sustraer el muestreo de las mediciones se desarrolló un programa que utiliza el hecho de que las variaciones entre los puntos pertenecientes a un mismo escalón son muy pequeñas, comparadas a las que tienen puntos vecinos que pertenecen a escalones diferentes. Además, usando el elemento de referencia situado al final de cada secuencia, el cual se observar en la figura 5.3b como el escalón más bajo, se puede relacionar cada escalón con su elemento correspondiente. Una vez se tiene esta información, el programa genera el valor para cada elemento sacando el promedio de los puntos correspondientes a él.

5.2. Fotografías obtenidas

En esta sección se presentan las fotografías obtenidas con dos métodos. El primero es un método simple y se encuentra resumido en la expresión 2.3. El segundo es sensado comprimido en el cual se hace uso de los modelos descritos en el capítulo 3. En total se utilizan 2 imágenes a las cuales se hace referencia como imagen 1, que corresponde a una letra “e”, e imagen 2 que corresponde a una letra “o”. Estas se obtienen de la impresión de un texto en acetato que se coloca en el arreglo experimental para realizar las proyecciones correspondientes a la superficie del DMD. En la figura 5.5 se puede observar la representación digital y redimensionada, obtenida con la cámara CMOS, de estas imágenes.

(a) (b) (c)

Figura 5.5: Imágenes utilizadas con el arreglo experimental obtenidas con la cámara CMOS y redimensionalizadas. En (a) se tiene la imagen 1 en dimensiones 64x64, en (b) la imagen 1 en dimensiones 32x32 y en (c) la imagen 2 en dimensiones 32x32

5.2.1. Resultados método simple

En esta sección, para obtener la imagen deseada, no se utiliza el sensado comprimido sino el método resumido en la expresión 2.3 que para recordar se presenta a continuación:

$$\vec{f} \simeq \frac{1}{m} \sum_{i=1}^{m} (y_i - \langle \vec{y} \rangle) \varphi_i$$

Donde $\vec{y}$ es el muestreo realizado y $\langle \vec{y} \rangle$ representa el promedio de los elementos de este.

El único procedimiento extra que se aplica al muestreo es restarle el promedio de todos sus elementos. Este método no utiliza las suposiciones del CS y consiste básicamente en sumar (combinación lineal) los elementos (renglones) de la matriz del muestreo pesados por un valor proporcional a su muestreo correspondiente. El objetivo de esto es poder tener este método como comparación para poder apreciar de forma más clara el desempeño que tiene el CS.

En este caso los porcentajes de muestreo son diferentes y corresponden a 10 %, 50 %, 100 %, 140 % y 180 %, donde los porcentajes mayores a 100 % significa que se realizaron más mediciones que elementos en la señal. Para esto se construyen matrices aleatorias de las dimensiones necesarias con la misma distribución Bernoulli. Se coloca la imagen 2 en el arreglo experimental y se obtienen los muestreos con el arreglo experimental.

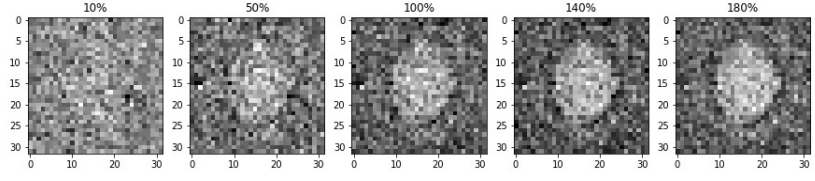


Figura 5.6: Reconstrucción de imagen 2 (5.5c) usando método resumido en la expresión 2.3 para 5 porcentajes de muestro diferentes.

En la figura 5.6 se observan las reconstrucciones obtenidas para los diferentes porcentajes de muestreo. Hay que mencionar que para generar estas imágenes se realiza un escalamiento al espacio de pixeles donde el mínimo valor de la reconstrucción se manda a 0, el máximo a 255 y los valores intermedios a puntos en este intervalo manteniendo las proporciones entre ellos. Aunque esto no entrega los valores verdaderos de la imagen permite tener una visualización clara de la estructura de la imagen que se busca representar.

Pasando a los resultados obtenidos se tiene que la reconstrucción es irreconocible para porcentajes menores a 100 % y para porcentajes mayores se empieza a distinguir de forma muy burda. El problema es que estos porcentajes ya no son convenientes ya que son superiores al 100 % y lo que se busca es obtener una representación comprimida de la señal, de otra forma se optaría por realizar un muestreo pixel por pixel.

5.2.2. Resultados del sensado comprimido

En esta sección se utiliza el método de sensado comprimido para obtener el muestreo de la imagen 1 y posteriormente los diferentes modelos para reconstruirla. Las reconstrucciones finales se representan de la misma forma que en la sección 5.2.1, esto es escalando los valores de la reconstrucción al intervalo [0,255]. Lo interesante en esta sección es que la reconstrucción final se obtiene en dos resoluciones. Una baja en la que las dimensiones finales son de 32x32 y otra alta con dimensiones de 64x64. Para lograr la última lo que se hace es dividir el área de muestreo en 4 bloques y cada uno de estos, que corresponde a 32x32 pixeles, se muestrea y reconstruye de forma independiente para al final juntarse y obtener la reconstrucción completa.

Para poder entender bien algunas de las características más relevantes de estos resultados, antes de proseguir a estos se tienen que aclarar un punto muy importante: Los muestreos obtenidos con el arreglo experimental no se pueden usar directamente con los modelos para obtener la reconstrucción de la imagen. Esto es consecuencia de dos factores. El primero es que cuando la matriz de muestreo se representa en los microespejos del DMD los elementos con valor -1 en realidad adquieren un valor 0 (no reflejan luz al detector) lo que hace que todos los valores del muestreo sean positivo en vez de estar centrado en 0

o muy cerca de este. El segundo va relacionado con la naturaleza de la imagen que se quiere muestrear. Para entrenar los modelos se obtuvieron muestreos de imágenes digitales que son discretas y cada punto (pixel) tiene un valor entre 0 y 255. Por otro lado el muestreo que se obtiene con el arreglo experimental es de una imagen continua, que se discretiza al reflejarse en el DMD y cuyos valores no viven en el intervalo [0,255] sino que dependen de los instrumentos de medición (osciloscopio y fotodetector). Más adelante se ahonda en estos dos puntos, pero lo que es de interés ahora es que debido a ellos se tiene que aplicar un preprocesamiento a los muestreos antes de poderlos usar con los modelos. En este trabajo se obtienen resultados para dos diferentes formas de preprocesamiento que se resumen en dos transformaciones denotadas como T_{teo} y T_{aprox} las cuales se describen en la sección 5.3, pero en pocas palabras el objetivo de ambas es centrar el muestreo en 0 o muy cerca de 0 y escalar los valores para simular que se obtuvieron a partir de puntos (pixeles) en el intervalo [0,255]. Ambos preprocesamientos dependen de dos parámetros los cuales se obtienen a partir del muestreo de una imagen proyectada en el DMD y una versión digital de esta misma imagen obtenida con la cámara CMOS (Esto se explica en la siguiente sección). La idea es que una vez se tienen estos parámetros se pueden usar para reconstruir imágenes a partir de muestreos sin la necesidad de la versión digital de la imagen. El resultado óptimo se puede lograr cuando para cada muestreo se obtiene también la imagen digital y se obtienen parámetros específicos. El problema aquí es que se está usando la imagen que se desea obtener para lograr esto por lo que no tiene ningún sentido como aplicación.

Los resultados obtenidos se dividen en 3 partes en las que cambia el tipo de preprocesamiento usado en los muestreos así como la forma de ajustar los parámetros que utiliza. A continuación se presentan estos resultados y las características de cada uno.

Reconstrucción ideal usando T_{teo}

Aquí lo que se busca es presentar las imágenes obtenidas usando la transformación T_{teo} de forma ideal, y por ideal se entiende que para cada muestreo se obtienen parámetros de escalamiento específicos en vez de usar los mismos valores para todos. Esto claramente no tiene mucho sentido como aplicación

ya que para poderlo hacer se necesita la versión digital de la imagen que se desea obtener. Sin embargo se realiza para poder tener un resultado "ideal" con el cual comparar el resto.

En las figuras 5.7 y 5.8 se pueden observar las reconstrucciones obtenidas con cada modelos para los 5 porcentajes de muestreo en las dos resoluciones anteriormente mencionadas (32x32 y 64x64). Lo primero que hay que notar, lo cual era un resultado esperado, es que conforme aumenta el porcentaje de muestreo la calidad de la reconstrucción también lo hace. Para el caso de dimensiones de 32x32 la estructura de la imagen resulta evidente hasta el porcentaje de muestreo de 20 % y mejora para el caso de 40 %. Por otro lado, para el caso de 64x64 a partir del muestreo de 10 % la estructura general de la imagen ya se percibe, en algunos modelos mejor que en otros, y para el caso de 40 % ya es realmente nítida. Otra observación que hay que hacer es que en el caso de 64x64 es claro como la reconstrucción se realiza con 4 bloques por separado ya que las fronteras de estos son notorias, aunque conforme el porcentaje de muestreo aumenta estas son cada vez menos bruscas.

Comparando el desempeño de los modelos en la parte visual, claramente el peor es stanNet. Por otro ladol, los modelos LinearNet, reconNet2 y DR2Net generaron las reconstrucciones más nítidas. Hay que recalcar que esta evaluación es desde un punto de vista visual. Los resultados formales se tratan más adelante.

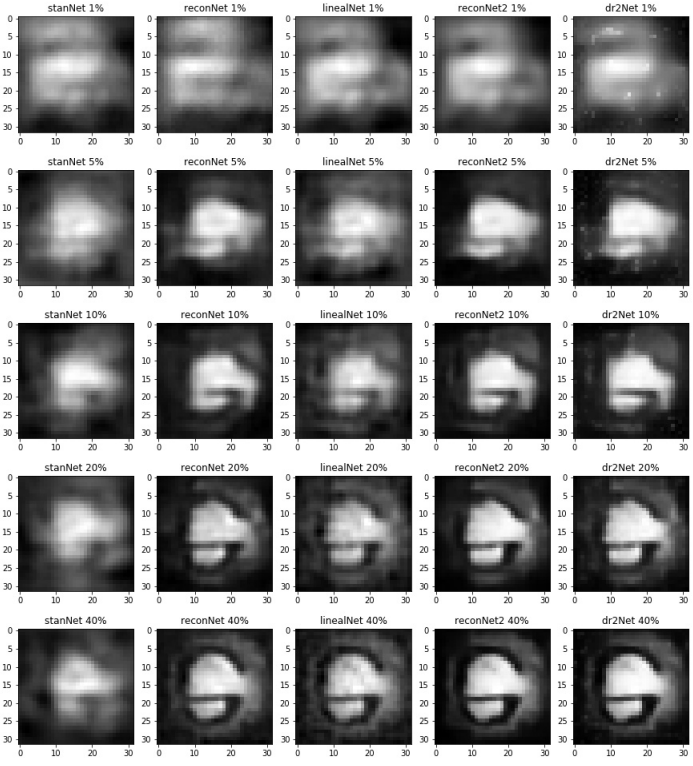


Figura 5.7: Reconstrucciones de imagen 1 en resolución 32x32 (5.5b) para los 5 porcentajes de muestreo y para los 5 modelos de forma adativa usando la transformación T_{teo} para el procesamiento de los datos experimentales

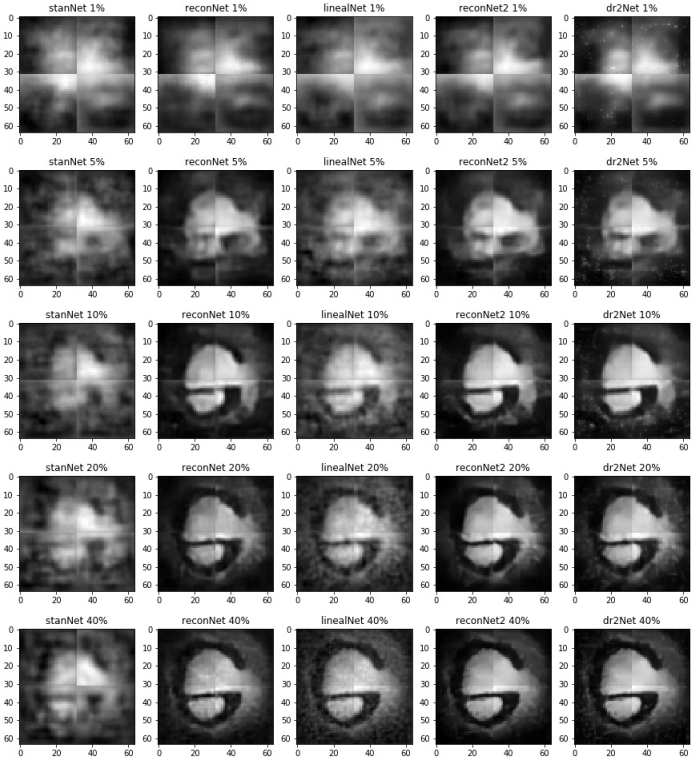


Figura 5.8: Reconstrucciones de imagen 1 en resolución 64x64 (5.5a) para los 5 porcentajes de muestreo y para los 5 modelos de forma adativa usando la transformación T_{teo} para el procesamiento de los datos experimentales

Reconstrucción real usando T_{teo}

Ahora se presentan las imágenes obtenidas usando la transformación T_{teo} de forma real, es decir, con los mismos parámetros de escalamiento para todos los muestreos. Estos parámetros se definen promediando los parámetros

ideales correspondientes obtenidos en los muestreos de 40 % para cada uno de los 4 bloques de la imagen de dimensiones 64x64. Una vez se han fijado estos parámetros se realizan las reconstrucciones de la imagen 1 para dimensiones de 64x64. Se escoge únicamente esta resolución ya que representa en realidad 4 imágenes diferentes, que son los 4 bloques en los que se divide la imagen 1, con lo que se puede realizar una evaluación mejor de los resultados.

En la figura 5.9 se tienen las reconstrucciones obtenidas. Como se observa hay un problema grave ya que algunos cuadrante no son reconstruidos de forma correcta lo que se debe a que los parámetros que están siendo utilizados para la transformación T_{teo} no son los correctos. Y aunque se puede argumentar que la imagen aún se puede distinguir hay que tener en cuenta que los parámetros fueron obtenidos para esta imagen en particular, si se usara una distinta muy probablemente el resultado sería peor.

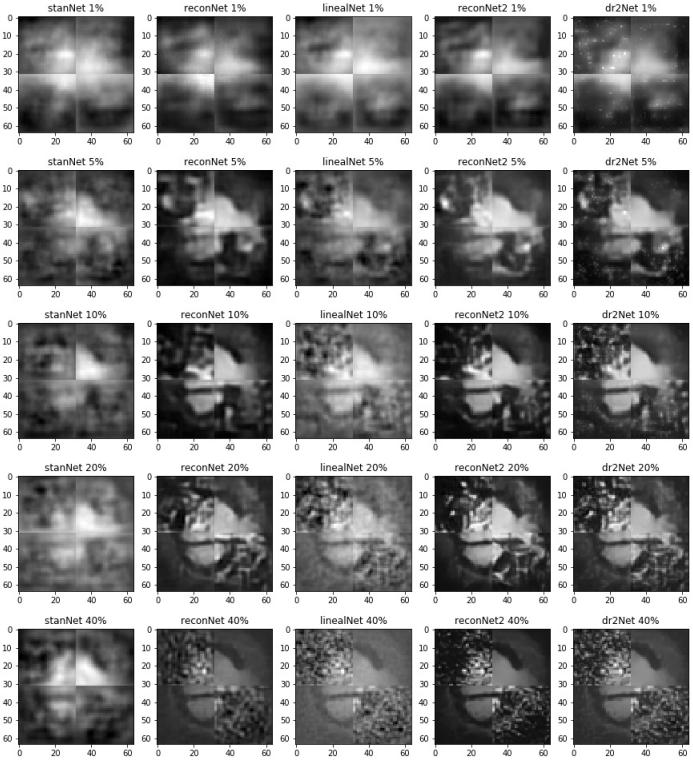


Figura 5.9: Reconstrucciones de imagen 1 en resolución 64x64 (5.5a) para los 5 porcentajes de muestreo y para los 5 modelos de forma no adaptiva usando la transformación T_{teo} para el procesamiento de los datos experimentales.

Reconstrucción real usando T_{aprox}

En esta última parte lo que se hace es cambiar el tipo de preprocesamiento que se realiza a los datos obtenidos del arreglo experimental, en vez de usar la transformación T_{teo} se utiliza T_{aprox}. Los valores de los parámetros de escala-

miento son fijos y se definen de la misma forma que en el caso real usando T_{teo}. Posteriormente se realizan las reconstrucciones para las dimensiones de 64x64. Las imágenes obtenidas se encuentran en la figura 5.10. En esta ocasión ya no se tienen el problema que se presenta al usar T_{teo}. Como se puede observar los 4 cuadrantes de las reconstrucciones tienen una nitidez bastante parecida y bastante cercana a la obtenida en el caso ideal (figura 5.8). La diferencia que se puede notar con respecto a este último, aunque poco perceptible, es que las imágenes obtenidas parecen tener un poco de ruido que se puede observar sobre todo en las regiones más lisas como por ejemplo el centro blanco.

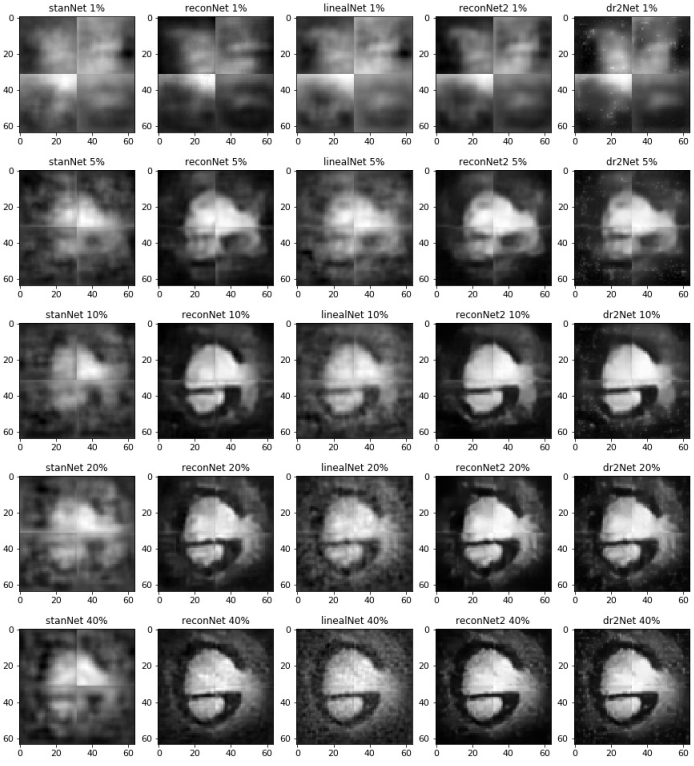


Figura 5.10: Reconstrucciones de imagen 1 en resolución 64x64 (5.5a) para los 5 porcentajes de muestreo y para los 5 modelos de forma no adaptiva usando la trnasformación T_{aporx} para el procesamiento de los datos experimentales.

5.3. Preprocesamiento de los datos

En las secciones anteriores se presentan las imágenes obtenidas. En el caso del sensado comprimido el proceso que se sigue es obtener el muestreo con el

arreglo experimental, realizar un preprocesamiento de estos muestreos, para lo cual se utilizan dos transformaciones: T_{teo} y T_{aprox}, y finalmente obtener la imagen reconstruida haciendo uso de los modelos. De esta forma se obtienen dos versiones reconstruidas de cada imagen, una para cada transformación. El resultado es que el preprocesamiento de T_{teo} no genera una imagen correcta mientras que T_{aprox} si lo hace.

En esta sección se describe con más detalle en qué consisten las transformaciones usadas para realizar el preprocesamiento de los muestreos. Además se explica el método que se ocupa para encontrar los parámetros del escalamiento y se lleva a cabo una pequeña discusión sobre como los valores que toman estos, en el caso de cada transformación, dan un indicio de porque T_{teo} no da buenos resultados.

Para poder explicar las transformaciones T_{teo} y T_{aprox} de forma concisa primero hay que entender la estructura general que comparten. Esta se puede dividir en dos pasos. Primero hacen un desplazamiento del muestreo y posteriormente lo escalan. Tomando esto en cuenta cada transformación se puede entender como la composición ($\circ$) de dos transformaciones. La que se encarga del escalamiento, y que es idéntica en ambos casos, se denota como R. Mientras que D_{teo} y D_{aprox} son las que se encargan del desplazamiento para T_{teo} y T_{aprox} correspondientemente. Las expresiones que resumen esto son las siguientes:

$$T_{teo} = R \circ D_{teo} \tag{5.1}$$

$$T_{aprox} = R \circ D_{aprox} \tag{5.2}$$

Desplazamiento del muestreo

Como se menciona anteriormente, el muestreo obtenido del arreglo experimental, al cual se hace referencia como muestreo físico, necesita ser desplazado debido a que la matriz con la que se obtiene no es la matriz de muestreo que se usa en el caso de imágenes digitales. Esta última está conformada con elementos 1 y -1 los cuales se representan como estados (ON/OFF) en el DMD. A los

elementos con valor 1 les corresponden el estado ON que reflejan la luz y por lo tanto es equivalentes. Sin embargo cuando los elementos tiene un valor -1 les corresponden microespejos con estado OFF que no reflejan luz "negativa" sino que la bloquean, teniendo así el efecto de un elemento con valor 0 en vez de -1. Esto quiere decir que el muestreo físico se genera con una matriz de muestreo con la misma estructura que la original pero que en las entradas con -1 más bien tiene 0. En el apéndice A se explica esto de forma más detallada y se llega a la siguiente transformación para obtener el desplazamiento correcto que se tiene que realizar al muestreo físico:

$$D_{teo}(\vec{y}_{fisico}) = 2 \cdot \vec{y}_{fisico} - ||\vec{x}||_{l_1} \tag{5.3}$$

Donde $\vec{y}_{fisico}$ es el muestreo del arreglo experimental y $\vec{x}$ es la señal o imagen.

Por otra parte se tiene la transformación D_{aprox}. A comparación D_{teo}, esta no es exacta sino una aproximación. Para poder entender de donde se obtiene primero hay que mencionar una característica de la matriz de muestreo. Debido a que esta se construye a partir de una distribución Bernoulli simétrica de elementos $\{1, -1\}$ cuando se realiza el muestreo de una imagen los valores de este estarán distribuidos de una forma más o menos uniforme alrededor de 0. Esto significa que si se toma el promedio de los puntos que conforman el muestreo este debe de ser cercano a 0. De hecho entre mayor sea el número de elementos en el muestreo más cercano a 0 debería ser este promedio. Esta idea es la que tiene como justificación la transformación D_{aprox} que lo que hace es centrar el muestreo obtenido con el arreglo experimental en cero y se puede resumir en la siguiente expresión:

$$D_{aprox}(\vec{y}_{fisico}) = \vec{y}_{fisico} - \frac{1}{N_{\vec{y}_{fisico}}} ||\vec{y}_{fisico}||_{l_1} \vec{1} \tag{5.4}$$

Donde:

- $\vec{y}_{fisico}$ es el muestreo del arreglo experimental
- $N_{\vec{y}_{fisico}}$ es la dimensión de $\vec{y}_{fisico}$
- $\vec{1}$ es un vector con 1 en todas sus entradas

Escalamiento del muestreo

Después de aplicar alguna de las transformaciones D_{teo} ó D_{aprox} a los datos del muestreo obtenido del arreglo experimental, falta un paso antes de poder utilizar los modelos. Este paso es necesario debido a una razón bastante simple: Los modelos fueron entrenados con muestreos generados a partir de imágenes representadas por pixeles que tienen un rango de valores posibles que vive entre 0 y 255 mientras que los muestreos hecho con el arreglo experimental vienen de imágenes representadas por voltajes que tienen un rango diferente. Lo que se necesita entonces es modificar la magnitud de los puntos del muestreo manteniendo la relación entre estos. La ventaja que se tiene es que en principio ambos instrumentos: cámara CMOS y fotodetector son proporcionales a la intensidad de las señales que detectan por lo que se puede suponer que la relación que se busca es lineal. Si se denota como $\vec{y_d}$ al muestreo obtenido de la imagen digital y $\vec{y_f}$ al obtenido del arreglo experimental después de aplicar alguna de las transformaciones de desplazamiento entonces se tiene la siguiente relación:

$$y_{di} = R(y_{fi}) = a \cdot y_{fi} + b \tag{5.5}$$

Donde el subindice i denota la entrada i-ésima del vector correspondiente.

La tarea que se tiene entonces es como encontrar los parámetros a y b correctos. Para esto se necesitan $\vec{y_d}$ y $\vec{y_f}$ de una imagen y con únicamente 2 entradas de cada uno el problema debería estar solucionado. Sin embargo hay dos detalles que pueden generar fallas. El primero es que $\vec{y_d}$ se obtiene de la imagen generada con la cámara CMOS la cual no corresponde exactamente a la imagen que se desea reconstruir, esto debido a que esta imagen tiene una dimensión mucho mayor a 32x32 o 64x64. Lo ideal sería obtener la imagen con el fotodetector pero para esto se tiene que reflejar la luz de los pixeles

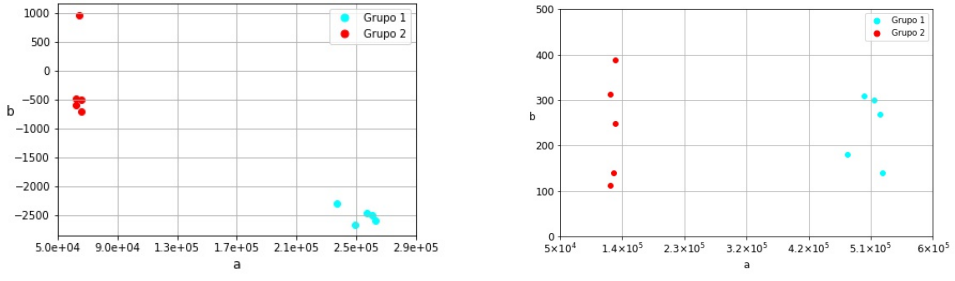
individuales, lo cual genera una señal muy débil que es prácticamente imperceptible. Debido a esto se usa la imagen obtenida con la cámara CMOS y se escala para tener al final una representación de ella de 32x32 o 64x64 pixeles (figura 5.5). El segundo detalle es el ruido que pueda tener $\vec{y_f}$. Por estas razones el método que se sigue para encontrar los parámetros, que se espera sea más robusto respecto a esta situación, consiste en minimizar la siguiente función definida a partir de los dos muestreos:

$$f(a,b) = \sum_{i=1}^{N} (y_{di} - (a \cdot y_{fi} + b))^2 \tag{5.6}$$

Hay que dejar claro que los parámetros a y b van a servir para relacionar los dos muestreos siempre y cuando el muestreo digital sea obtenido con la cámara CMOS en el mismo estado. Si este cambia, por ejemplo modificando el tiempo de exposición, se tienen que encontrar nuevos parámetros.

Análisis del escalamiento lineal

Hasta ahora aunque todo parece estar bien, esto depende de que la relación 5.5 sea correcta. Para dar una desmostración sobre esto se lleva a cabo un experimento simple. Supóngase que se tienen 5 imágenes diferentes y se

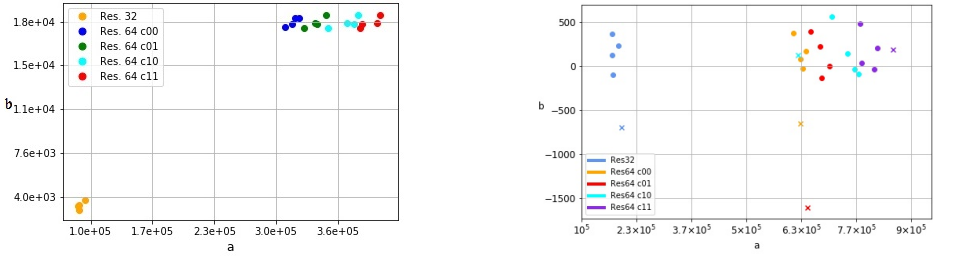


(a) (b)

Figura 5.11: Parámetros de relación lineal (expresión 5.6) obtenidos para imágenes de Grupo1 y Grupo 2. En la figura (a) se obtuvieron para la transformación T_{teo} y en la figura (b) para T_{aprox}

obtienen los muestreos $\vec{y_d}$ y $\vec{y_f}$ para cada una de ellas cuidando que no se cambie el estado de la cámara CMOS ni la iluminación en el arreglo experimental. Entonces los parámetros a y b obtenidos para cada uno de estos 5 muestreos deberían ser iguales o muy parecidos. Además si se repitiera esto para otro conjunto de 5 imágenes pero ahora con un estado diferente en la cámara, los nuevos parámetros a y b obtenidos deberían ser diferentes a los anteriormente comentados. Esto se realiza para dos grupos de imágenes, a los cuales se hace referencia como Grupo 1 y Grupo 2, para el caso de muestreo de 10 %. Los parámetros a y b obtenidos para las dos transformaciones T_{teo} y T_{aprox} se encuentran en la figura 5.11. Lo primero que se puede observar es que efectivamente los parámetros de los muestreos de un grupo son parecidos entre sí pero entre grupos diferentes presentan variaciones fuertes lo que se entiende como una consecuencia de la relación lineal existente.

Otra observación bastante interesante que se puede hacer es con respecto a las magnitudes de los parámetros para cada transformación. Mientras que para el caso de T_{teo} la magnitud de b entre grupos tiene una variación de un orden de magnitud en el caso de T_{aprox}, esta prácticamente no existe. Para ver si esto es algo que se cumple siempre o fue casualidad lo que se hace es encontrar los parámetros a y b para las dos transformaciones y todos los muestreos (porcentajes 1 %, 5 %, 10 %, 20 % y 40 %) de la imagen 1 en dimensiones 64x64 (figura 5.5a). Estos resultados se pueden observar en la figura 5.12. El resul-



(a) (b)

Figura 5.12: Parámetros de relación lineal (expresión 5.6) obtenidos para los porcentajes de muestreo de 1 % ,5 %, 10 %, 20 % y 40 % para la imagen 2 con dimensiones de 64x64. En la figura (a) se obtuvieron para para la transformación T_{teo} y en la figura (b) para T_{aprox}. Además en esta última los correspondientes al caso de 1 % aparecen con cruces en vez de puntos

tado es claro, no se trata de una casualidad y además esta vez la magnitud de b para la transformación T_{teo} es mucho más grande, incluso comparable a la de a. Por otro lado para T_{aprox} esta es pequeña. Si bien hay algunos puntos de magnitud mayor estos corresponden a marcadores en forma de "x" lo cual significa que corresponden al muestreo de 1 %. Lo particular de estos casos es que sus muestreos cuentan únicamente con 10 elementos y por lo tanto tienen una gran posibilidad de que exista un sesgo en el cálculo de a y b. Entre más grande es el número de elementos en el muestreo esta probabilidad disminuye y los resultados son más confiables.

Los resultados encontrados para el parámetro b que marcan la diferencia entre las transformaciones T_{teo} y T_{aprox} son una prueba de porque los resultados de la primera no funcionan y de la segunda sí. Para esto primero hay que entender que la diferencia entre ellas se resume en D_{teo} y D_{aprox} que tienen como objetivo desplazar el muestreo físico. Esto quiere decir que idealmente al aplicar la transformación R (5.5) el parámetro b debería ser cero y lo único que se debería hacer es escalar los puntos del muestreo. Es muy claro entonces que en T_{teo} algo está mal, ya que los valores de b son grandes lo que no sucede en T_{aprox}. Esto probablemente se deba a que D_{teo} (expresión 5.3) no toma en cuenta que puede existir ruido en el muestreo o que el fotodetector o la cámara CMOS tienen un desplazamiento al interpretar la señal que reciben.

Capítulo 6

Desempeño de los modelos

En el capítulo anterior se presentan los resultados obtenidos a partir del arreglo experimental y el uso de los modelos y se discute el procedimiento que se necesita para llevar esto a cabo. Ahora lo que se busca es centrarse un poco más en los modelos. Claramente estos tienen desempeños diferentes y calificar esto únicamente desde un punto visual puede ser complicado. En este capítulo entonces se hace una comparación más formal entre los modelos. Para la primera parte esta comparación se realiza con imágenes digitales además de que se comentan algunos aspectos importantes sobre estas que tienen repercusiones fuertes en la parte visual de las reconstrucciones obtenidas en el capítulo anterior. Posteriormente se analizan los muestreos que se obtienen con el arreglo experimental para caracterizar el ruido de estos y una vez se tiene esta información se puede ver cómo reaccionan los modelos a este tipo de ruido. Al final se comparan también formalmente las reconstrucciones obtenidas usando el arreglo experimental.

Para poder tener una idea de la calidad que se obtiene para una reconstrucción hay que introducir la Proporción Máxima de Señal a Ruido o PSNR que comúnmente se emplea para este fin y se define de la siguiente forma:

$$PSNR = 20 \cdot \log_{10}(\frac{MAX_I}{\sqrt{ECM}}) \tag{6.1}$$

Donde ECM es el error cuadrático medio entre la reconstrucción y la ima-

gen y MAX_I es el valor máximo que puede tomar un pixel que en este caso des 255.

Teniendo el PSNR se puede entonces realizar comparaciones mucho más claras entre los diferentes modelos.

6.1. Comparación de modelos con imágenes digitales

El sensado comprimido, como se explica en el capítulo 2, es un método cuyo desempeño depende de ciertas suposiciones, las cuales al verse modificadas pueden influir de forma drástica en su desempeño. La suposición más importante es sin duda la dispersión de la señal. Entre más dispersa sea esta, mejor debería ser la reconstrucción obtenida. Hay que mencionar que las imágenes naturales son consideradas señales dispersas ya que tienen áreas extensas con colores o texturas más o menos constantes, esto en términos más técnicos significa que la variación en los valores de pixeles vecinos es pequeña en general. Esto se puede conectar entonces con la resolución de las imágenes. Una mayor resolución, que significa una mayor densidad de pixeles por unidad de área, implica entonces una mayor dispersión de la señal. Tomando en cuenta esto surge una duda. Y es que si se piensa en el conjunto de imágenes con el que se entrenaron los modelos, probablemente no sea tan disperso como uno de imágenes naturales hablando en términos de resolución. Esto debido a que son imágenes que fueron escaladas a un tamaño de 32x32 pixeles lo que puede hacer que los valores de pixeles vecinos tengan grandes variaciones. Esto da pie a dos cuestiones sumamente importantes y que están relacionadas entre sí. La primera concierne al entrenamiento de los modelo y más específicamente a los datos con los que se realiza este. Como ya se mencionó estos datos son imágenes que probablemente no sean tan dispersas como otras de mayor resolución y la cuestión es averiguar si los modelos entrenados son capaces de generalizar el proceso de CS a imágenes más dispersas. Esta idea da pie a la segunda cuestión y es que tomando en cuenta la suposición de la dispersión, esto debería implicar que entre más dispersa sea una imagen mejor debería ser su calidad de reconstrucción. Para poder contestar estas preguntas y al mismo tiempo realizar una comparación entre los modelos se tiene la figura

6.1. Para esta figura se usaron tres conjuntos de imágenes. El primero fue el conjunto CIFARtest que se usa para validar los modelos y cuenta con 10 000 elementos. El segundo cuenta con 11 654 elementos que se obtuvieron al dividir una fotografía digital de dimensiones 3456x3456 en bloques de 32x32. Y el tercero es un conjunto de 5 000 elementos que corresponde al set de validación de COCO [12] y cuyas imágenes fueron reducidas a un tamaño de 32x32. El proceso que se sigue para obtener los valores que se observan en la figura es muestrear todas las imágenes de cada conjunto para cada porcentaje y realizar la reconstrucción de cada una usando todos los modelos. Posteriormente obtener el PSNR de cada reconstrucción y finalmente obtener el PSNR promedio de cada conjunto por porcentaje. Lo que se puede observar inmediatamente es que la calidad de la reconstrucción para cada conjunto tiene variaciones considerables. Como se esperaba el conjunto compuesto por elementos de la imagen real tiene una calidad promedio de reconstrucción mucho más alta que los otros 2 conjuntos, superándolos mínimo por 8 puntos de PSNR lo que representa una gran diferencia. Por otro lado hay algo que se tiene que notar que no es muy intuitivo y es que entre el conjunto CIFARtest y COCO también existe una diferencia, aunque no tan notable como con el conjunto de la imagen real. Al tener únicamente la información de estos dos conjuntos se podría pensar que su calidad de reconstrucción debería resultar muy similar ya que ambos son imágenes naturales reducidas a dimensiones de 32x32, sin embargo el resultado es que la calidad del conjunto CIFARtest es mejor. La respuesta más probable a este comportamiento es que el conjunto CIFARtest se usa como validación a la hora del entrenamiento de los modelos, esto quiere decir que las instancias finales seleccionadas para cada uno de estos se escogen a partir de este conjunto. Por esta razón el desempeño para CIFARtest puede ser particularmente mejor que el de cualquier otro conjunto. En este sentido el desempeño real que se espera de los modelos debería tomarse como el obtenido para los otros dos conjuntos.

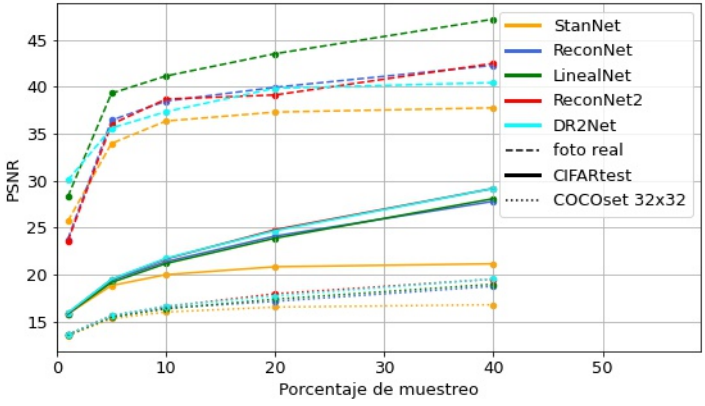


Figura 6.1: Comparación de la calidad promedio de reconstrucción, medida como PSNR, para 3 conjuntos de imágenes para los 5 modelos y diferentes porcentajes de muestreo. En las leyendas los colores (exceptuando el negro) denota el modelo que se utiliza mientras que el tipo de línea (en negro) el conjunto al cual se aplica el modelo.

En términos generales lo que se puede ver en la figura 6.1 es que para el conjunto COCO de resolución pequeña (32x32) los modelos que mejor desempeño tienen son ReconNet2 y DR2Net y la diferencia con los otros modelos no es muy notoria, llegando a 3 puntos de PSNR respecto a StanNet que fue el peor. Para el caso del conjunto de gran resolución (Imagen real) las cosas cambian. En este caso el modelo LinearNet tuvo el mejor desempeño seguido de los modelos ReconNet y ReconNet2 para todos los porcentajes de muestreo excepto para 1 %, para este los últimos dos modelos son mejores que LinearNet por un pequeño margen. Esta vez las diferencias en la calidad de reconstrucción son más notorias sobre todo conforme el porcentaje de muestreo crece. Para el caso de 40 % por ejemplo, el modelo LinearNet supera por alrededor de 5 puntos de PSNR a los que le siguen y por más de 10 a StanNet que nuevamente tuvo el peor desempeño. La otra cosa importante a notar es que la primera capa de los modelos ReconNet2 y DR2Net se inicializa con los pesos del modelo LinearNet antes de comenzar el entrenamiento. Esto implica que la estructura extra que tienen está repercutiendo de forma negativa más que ayudando.

La comparación que se hace para los conjuntos anteriores, además de per-

mitir hacer una primera comparación entre los modelos, parece confirmar la importancia de la dispersión de las imágenes, relacionada con su resolución, para la calidad final de su reconstrucción. Sin embargo, esta última comparación se hizo con conjuntos que contienen diferentes imágenes lo cual no es lo ideal. Por esta razón, y para poder corroborar lo observado de una forma más clara, se realiza lo siguiente: se escogen 5 fotografías de resolución 3456x3456, a las cuales se hará referencia como foto 1, foto 2, foto 3, foto 4 y foto 5. Vale la pena mencionar que estas fotografías fueron seleccionadas en función de que tan simples son, y por simples se refiere a que tan grandes son las variaciones de los valores de los pixeles vecinos en ella. Las fotos 1 y 5 son intermedias, las fotos 2 y 4 tienen pocas variaciones mientras que la foto 3 tiene muchas. Hay que tener en cuenta que esta "clasificación" de simpleza se hizo únicamente de forma visual. Estas fotografías se pueden observar en la figura 6.2. A partir de ellas se crearon 13 versiones de cada una. Lo que cambia en cada versión es la resolución, lo que se logra variando sus dimensiones las cuales se escogen como: 32x32, 288x288, 576x576, 864x864, 1152x1152, 1440x1440, 1728x1728, 2016x2016, 2304x2304, 2592x2592, 2880x2880, 3168x3168 y 3456x3456. Se hacen entonces 13 conjuntos para cada fotografía dividiendo su versión correspondiente en bloques de 32x32. El primero por ejemplo tiene únicamente un bloque de dimensiones 32x32. El segundo tiene 81 bloques de 32x32 que conforman la fotografía de dimensión 288x288. Y así con los demás conjuntos. Una vez los conjuntos conformados, se realiza el mismo procedimiento que en la figura 6.1 para obtener el PSNR promedio por conjunto para cada modelo. Los datos obtenidos son demasiados para presentarlos de una forma concisa y se pueden encontrar en forma de tabla en el apéndice B. Si se observan los datos en la tabla los mejores modelos fueron en general DR2Net y Linear-

(a) (b) (c) (d) (e)

Figura 6.2: Fotografías reales utilizadas para entender la relación entre calidad de reconstrucción y resolución. De izquiera a derecha se tienen: Foto 1, foto 2, foto 3, foto 4 y foto 5

Net en escenarios distintos que se comentan en breve. ReconNet2 muestra resultados muy parecidos a los de DR2Net mientras que nuevamente StanNet tiene los peores resultados. Debido a que el interés se tiene en el mejor modelo el análisis que se hace a continuación se realiza para DR2Net y LinearNet.

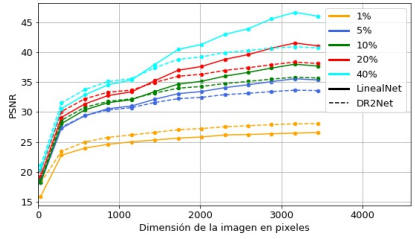


Figura 6.3: Calidad de reconstrucción promedio medida como PSNR hecha con LinearNet y DR2Net para la imagen 1 en función de su dimensión (o resolución). Los resultados se muestran para 5 porcentajes de muestreo: 1 %, 5 %, 10 %, 20 % y 40 %

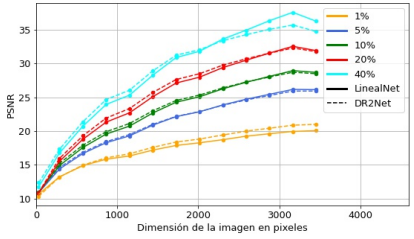


Figura 6.4: Calidad de reconstrucción promedio medida como PSNR hecha con LinearNet y DR2Net para la imagen 1 en función de su dimensión (o resolución). Los resultados se muestran para 5 porcentajes de muestreo: 1 %, 5 %, 10 %, 20 % y 40 %

En las figuras 6.3 Y 6.4 se tiene el PSNR promedio en función de las dimensiones de la imagen muestreada, esto para las fotografías 2 y 3 correspondientemente. Primero se analiza la figura 6.3 en donde se usa la fotografía 2 la cual está clasificada como simple. La primera conclusión, que es muy clara, es que efectivamente conforme aumenta la resolución de la fotografía la calidad de la reconstrucción también lo hace y esto es independiente del porcentaje de muestreo. Por otra parte se tiene el comportamiento de los 2 modelos utilizados el cual es un poco más complicado y esta vez sí depende del porcentaje de muestreo. Para el muestreo de 1 % el modelo DR2Net es siempre mejor, independientemente de la resolución. Sin embargo para los demás porcentajes algo particular pasa. Para dimensiones menores a 1500x1500 el desempeño de DR2Net es en general mejor aunque por un margen pequeño. Pero para dimensiones mayores el modelo LinearNet comienza a tener un desempeño considerablemente superior lo cual se vuelve más marcado conforme el porcentaje de muestreo aumenta. Por ejemplo para la dimensión de 3456x3456 muestreando el 40 % de la señal la diferencia en la calidad llega a ser de alrededor de 5 puntos de PSNR. Ahora se pasa a la figura 6.4 en donde se usa la fotografía 3 que está clasificada como compleja. El comportamiento general

respecto al de la fotografía 2 no cambia, pero lo que si pasa es que las diferencias entre los dos modelos disminuyen drásticamente. Para los porcentajes de muestreo de 5 %, 10 % y 20 % el desempeño es muy cercano. En donde existe una diferencia mayor es en el caso de 1 % donde DR2Net es mejor y en 40 % en donde LinearNet si es significativamente mejor. Una última observación que se obtiene de comparar las dos figuras es que en general el desempeño de los modelos es sin lugar a dudas mejor para la fotografía 2 que para la 3, lo que de alguna forma confirma que la "clasificación" de simpleza que se hizo tiene sentido.

En resumen lo que se puede concluir es lo siguiente. Primero, a mayor resolución mejor calidad de reconstrucción. Hay que hacer énfasis en que esto es para la resolución. En las figuras 6.3 y 6.4 se mide en función de la dimensión pero para la misma imagen, cuando se disminuye el tamaño de una imagen lo que se está haciendo es disminuir su resolución. Segundo, cuando el porcentaje de muestreo es pequeño el modelo DR2Net, seguido muy de cerca por ReconNet2, tiene el mejor desempeño, sin embargo si se usan porcentajes más grandes el modelo LinearNet puede entregar resultados superiores para resoluciones grandes.

Para poder tener una idea más clara de cuál es el desempeño del sensado comprimido con imágenes de diferente resolución se presenta un resultado visual. Para esto se utiliza una imagen de gran resolución la cual tiene las si-

(a) (b)

Figura 6.5: Imagen de dimensiones 4608x3456. En (a) se muestra la imagen completa mientras que en (b) se muestra la región delimitada por el cuadrado verde que tiene dimensiones de 125x125.

guientes dimensiones: 4608x3456. Esta imagen se puede observar en la figura 6.5a, además en la figura 6.5b se observa un aumento a la región delimitada por el recuadro verde en la figura 6.5a. Esta región tiene dimensiones 125x125. Lo que se hace es obtener el muestreo de la imagen para los 5 porcentajes, dividiéndola en bloques de 32x32, y posteriormente obtener las reconstrucciones de cada uno usando el modelo LinearNet que fue el que tuvo el mejor desempeño para este tipo de imágenes. En la figura 6.6 se pueden observar los resultados obtenidos para cada porcentaje de muestreo. En la columna izquierda se tiene la reconstrucción completa mientras que en la derecha se tiene la reconstrucción correspondiente a la región en la figura 6.5b. Lo que se puede observar es que para la imagen completa incluso el porcentaje de muestreo del 1 % permite distinguir la estructura general. Sin embargo si hacemos un aumento en regiones más pequeñas, lo que se tiene en la columna derecha de la figura 6.6, se puede notar claramente la diferencia entre los distintos porcentajes de muestreo. Para los casos de 1 % y 5 % la región aumentada no se puede distinguir y además las fronteras entre los bloques son muy notorias. Para el caso de 10 % la estructura de la región se puede empezar a distinguir y las fronteras de los bloques son menos bruscas. Y finalmente para los muestreos de 20 % y 40 % la estructura se distingue de forma clara y las fronteras entre los bloques son prácticamente imperceptibles.

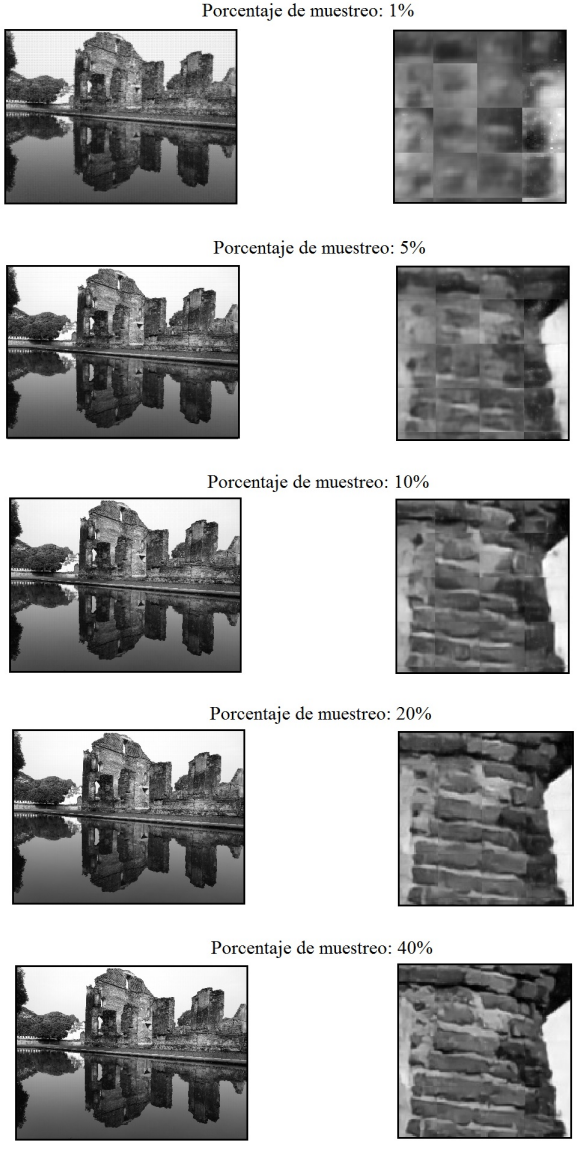


Figura 6.6: En la columna izquierda se tienen las reconstrucciones de la imagen en la figura 6.5a hechas con el modelo DR2Net. En la columna izquierda se tiene un aumento de la región delimitada por el rectangulo verde en la misma figura para la reconstrucción correspondiente.

6.2. Señales con ruido

En esta sección se buscan dos cosas. La primera es caracterizar el ruido presente en los muestreos obtenidos con el arreglo experimental. La segunda es simular este ruido de forma digital y ver el desempeño de los modelos para este tipo de datos. De esta forma se puede entender un poco más las reconstrucciones obtenidas que son presentadas en la sección 5.2.

Caracterización de ruido

En un experimento siempre hay que tener en cuenta que los datos obtenidos pueden tener ruido. En el caso particular de este trabajo factores como fuentes de luz externas al experimento, la precisión de los instrumentos de medición y otros, son los responsables de generar el ruido del que se habla. Algo que es de interés, teniendo en cuenta lo anterior, es cuánto ruido se tiene en las mediciones y que tipo de ruido es, de esta forma se puede conocer que tan relevantes pueden ser los cambios que se generen por su influencia. El tipo de ruido en el que se centra el análisis es relativo a cada elemento del muestreo. Para esto se usan los muestreos obtenidos para la imagen 1 (figura 5.5) con el arreglo experimental. Para está imagen se realizan muestreos en dos resoluciones: 32x32 pixeles y 64x64 pixeles, la última de estas está compuesta por el muestreo de 4 bloques de 32x32. Los muestreos en cada caso se realizaron para los 5 porcentajes de información que son: 1 %, 5 %, 10 %, 20 % y 40 %.

	μ	Std
32x32	0,0	209,34
64x64 c00	0,0	429,10
64x64 c01	0,0	407,20
64x64 c10	0,0	410,74
64x64 c11	0,0	422,24

Cuadro 6.1: Parámetros de caracterización de ruido intrínseco de los datos del osciloscopio

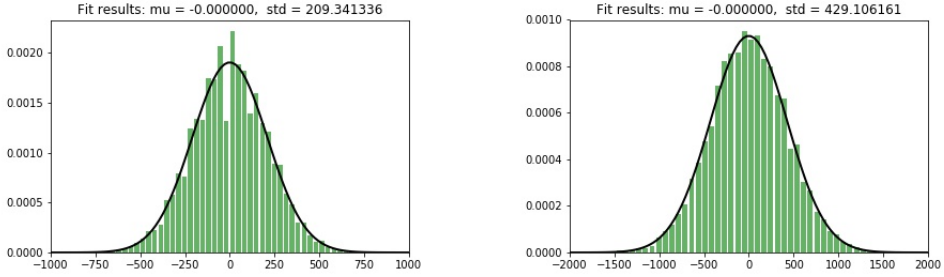
Lo que se hace para poder cuantificar el error es lo siguiente. Primero se seleccionan los datos obtenidos con el osciloscopio para el muestreo de 1 % en resolución de 32x32 pixeles. Después se toman únicamente los puntos que se

usan para obtener el promedio que corresponde al valor del primer elemento del muestreo, se les aplica la transofrmación T_{aprox} (es la que dio mejores resultados), y se centran en 0. Los valores resultantes se guardan. Esto se realiza para cada elemento del muestreo y para cada porcentaje de muestreo. Todos los puntos resultantes se juntan y guardan. El mismo proceso se realiza para cada uno de los 4 bloques que conforman el caso de resolución de 64x64 pixeles. De esta forma se obtienen 5 conjuntos de puntos todos centrados en cero. El ruido para cada conjunto se puede caracterizar entonces con el valor de su desviación estándar.

La sospecha que se tiene es que el ruido generado a la hora de tomar las mediciones sigue una distribución normal. Esto debido a que en el arreglo experimental no se cuenta con ninguna fuente de luz extra e intensa que pueda estar sesgando los valores de alguna otra forma. Para verificar esto lo que se hace es obtener un histograma normalizado para cada uno de los 5 conjuntos que se tienen y graficar la distribución normal generada a partir de su promedio y desviación estandar. En la figura 6.7 se pueden observar los histogramas del conjunto de resolución 32x32 y del cuadrante superior izquierdo del caso de resolución 64x64. Lo primero que salta a la vista es que el ajuste es bastante cercano lo cual confirma la sospecha y nos permite asegurar con evidencia que en efecto el ruido generado sigue una distribución normal. Esto es igual para los 3 conjuntos restantes para los cuales no se muestra la visualización gráfica pero en la tabla 6.1 se puede observar el parámetro μ, que es donde está centrada la distribución, y la desviación estándar (Std) correspondiente de cada uno de ellos.

Desempeño de modelos con ruido

Lo que se busca ahora es determinar cómo se ven afectados los modelos cuando los muestreos tienen un cierto nivel de ruido. Siguiendo el análisis que se hizo anteriormente sobre el ruido presente en los muestreos experimentales, se sabe que este sigue una distribución normal centrada en 0. En la tabla 6.1 se pueden encontrar los valores de las desviaciones estándar encontradas para 5 conjuntos suponiendo que se utiliza T_{aprox} para realizar el preprocesamiento.



(a) (b)

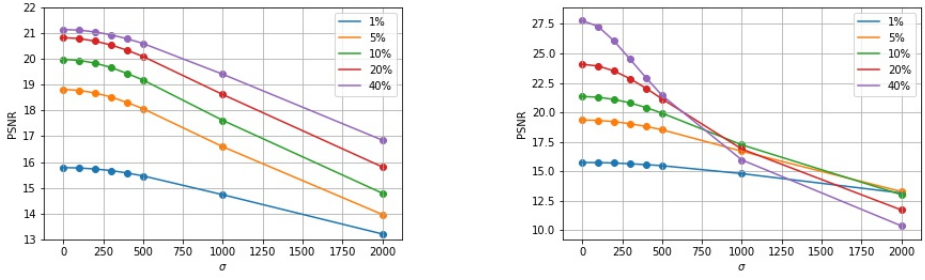
Figura 6.7: Histogramas de la distribución normalizada de los errores en la señal del osciloscopio. Para cada punto perteneciente a un escalón se obtuvo su distancia a su valor promedio y esto se realizó para todos los puntos de los 5 porcentajes de muestreo. En la figura (a) se observa el histograma de estos valores correspondiente a la imagen 2 en resolución 32x32. En la figura (b) el correspondiente al cuadrante c00 de la misma imagen en resolución 64x64. En ambos casos se tiene también la curva correspondiente a la distribución normal que mejor ajusta los datos.

Lo que se hace a continuación es replicar de forma digital el ruido y probar el desempeño de los modelos cuando este está presente en diferentes magnitudes. Siguiendo esta idea y basándose en los valores que se obtuvieron para las desviaciones estándar del ruido se generan 8 grupos de muestreos con diferentes niveles de ruido. Estos 8 grupos son generados a partir del conjunto CIFARtest y el procedimiento que se sigue es el siguiente:

- Escoger la desviación estándar para cada uno de los 8 grupos, las cuales son: 0, 100, 200, 300, 400, 500, 1000 y 2000
- Generar el muestreo de todos los elementos del grupo CIFARtest para cada porcentaje: 1 %, 5 %, 10 %, 20 % y 40 %
- Para construir el grupo con $\sigma = 100$ por ejemplo, lo que se hace es a cada elemento de cada muestreo sumarle un escalar generado con una distribución normal centrada en cero con desviación estándar igual a σ.

De esta forma se construyeron los 8 grupos con diferentes niveles de ruido.

Posteriormente se realizan las reconstrucciones de los muestreos de cada uno de estos grupos con cada modelo y se obtiene el PSNR promedio de cada

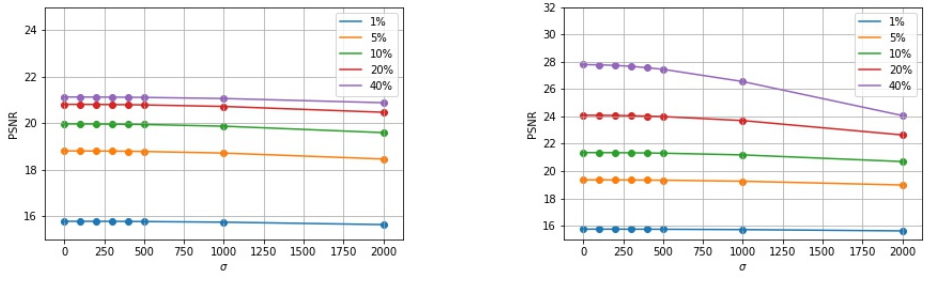


(a) (b)

Figura 6.8: PSNR promedio obtenido para los 8 grupos de ruido del conjunto CIFARtest para los 5 porcentajes de muestreo. En (a) se tiene los resultados obtenidos para el modelo stanNet mientras que en (b) los obtenidos para el modelo reconNet

uno. Esto genera una cantidad considerable de datos que se puede revisar en el apéndice C. Lo que es de interés en esta sección es encontrar cómo afecta de forma general el ruido a las reconstrucciones en relación a la magnitud de este. Lo que se hace entonces es analizar los resultados y obtener el comportamiento general e ilustrarlo con algunos de los datos más representativos. Esto se puede ver en la figura 6.8 en donde se escogen los resultados para dos modelos. En la figura 6.8a stanNet y en la figura 6.8b reconNet, más adelante se explica el porqué. Lo primero que salta a la vista, y era de esperarse, es que conforme la magnitud del ruido aumenta, independientemente del porcentaje de muestreo, la calidad de la reconstrucción disminuye. Este es el comportamiento general que se esperaba y es el mismo para los modelos que no se han mostrado.

Algo que llama la atención de los resultados obtenidos, es un comportamiento diferente para los distintos porcentajes de muestreo entre el modelo stanNet y los demás. Esta fue la razón de escoger los datos en las dos gráficas presentes en la figura 6.8. Para el modelo stanNet se observa la disminución de la calidad de reconstrucción conforme la magnitud del ruido aumenta pero además, independientemente del nivel de ruido un mayor porcentaje de muestro va a implicar una mejor calidad de reconstrucción. En el resto de los modelos este último comportamiento cambia. Esto lo podemos observar en la figura 6.8a para el caso de reconNet que sirve para ilustrar lo que sucede.



(a) (b)

Figura 6.9: PSNR promedio obtenido para los 8 grupos de ruido del conjunto CIFARtest para los 5 porcentajes de muestreo obteniendo el muestreo final del promedio de 37 mediciones. En (a) se tiene los resultados obtenidos para el modelo stanNet mientras que en (b) los obtenidos para el modelo reconNet

Se puede observar como a partir de una cierta magnitud de ruido un mayor porcentaje de muestro no genera una reconstrucción de mayor calidad y de hecho en algunos casos esto se vuelve lo contrario: Un mayor porcentaje de muestreo genera una peor calidad de reconstrucción. Este fenómeno es bastante interesante y se podría intentar ver más a fondo porque sucede en algún trabajo futuro.

Por último se vuelve a realizar el procedimiento para obtener los muestreos con ruido pero ahora cada uno se obtiene del promedio de 37 muestreos con el ruido gaussiano correspondiente. Esto se hace ya que es el procedimiento que se está siguiendo para los muestreos con el arreglo experimental. En estos se obtienen una serie de puntos con el osciloscopio y del promedio de estos se obtiene un valor del muestreo. El número de puntos que se usa para obtener el promedio, que es 37, se escoge porque es el número promedio de puntos que se obtiene con el osciloscopio para cada valor de muestreo cuando el DMD corre a la frecuencia más alta. En el caso de la frecuencia más baja este número asciende a 150. En este caso los resultados se obtuvieron únicamente para stanNet y reconNet y se pueden observar en la figura 6.9.

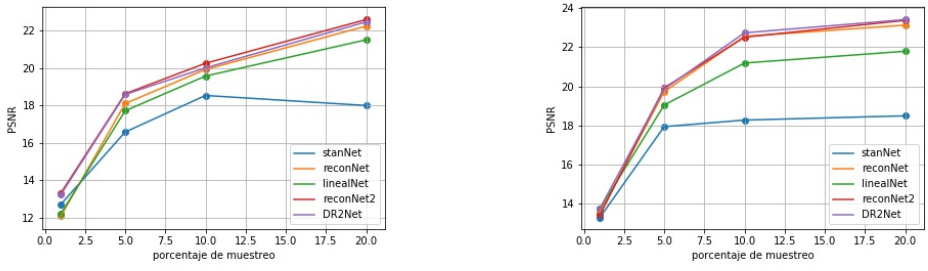
Comparando las figuras 6.8 y 6.9 Es evidente como el construir los muestreos a partir del promedio de varias mediciones puede disminuir mucho el

efecto negativo que tiene el ruido sobre la calidad de las reconstrucciones finales. En el caso particular del arreglo experimental usado en donde la desviación estándar del ruido se sitúa entre $\sigma = 200$ y $\sigma = 400$, y el número promedio de puntos usados para construir los valores de los muestreos es de al menos 37, es claro que el efecto del ruido no genera una pérdida considerable en la calidad de las reconstrucciones por lo que se espera que estas sean parecidas a las obtenidas de forma digital.

6.3. Desempeño formal de modelos

En esta última sección lo que se busca es calcular la calidad de las imágenes obtenidas a partir de los muestreos hechos con el arreglo experimental. Sin embargo para hacer esto hay un pequeño inconveniente y es que no se cuenta con la imagen exacta que se busca reconstruir. Esto se debe a que para obtenerla se necesita tomar el valor de cada pixel, lo que se logra reflejando únicamente la luz correspondiente a este en el DMD lo cual genera una señal muy pequeña que es prácticamente imperceptible para el fotodetector. La buena noticia es que aunque no tenemos la imagen exacta si tenemos una aproximación de esta. Los muestreos de 40 % generan una reconstrucción bastante cercana a la real, como se ha visto a lo largo de los resultados, especialmente algunos modelos como DR2Net. Lo que se hace entonces es tomar la reconstrucción realizada con el muestreo de 40 % por el modelo DR2Net y calcular la calidad de las reconstrucciones de los otros porcentajes de muestreo con respecto a esta. Esto no es lo ideal pero permite tener un resultado cercano. Además cuando se trabaja con cámaras fotográficas comerciales este es muchas veces el caso. Este tipo de cámaras guardan las imágenes obtenidas en formatos donde se pierde información, como JPEG, para que estas no sean tan pesadas.

Los resultados de la calidad obtenidos para la reconstrucción de la imagen 2 se presentan en la figura 6.10 para las dimensiones de 32x32 y 64x64. Como se puede observar la calidad medida como PSNR se mantiene en los niveles esperados y es ligeramente mayor para la imagen de mayor resolución lo que es congruente con lo obtenido en el caso digital.



(a) (b)

Figura 6.10: Calidad de reconstrucción medida en PSNR de imagen 1 en sus dos resoluciones (32x32 y 64x64) para los 5 modelos usando T_{aprox} como técnica de procesamiento. En (a) se tiene el caso de 32x32 y en (b) de 64x64.

Capítulo 7

Conclusiones

En esta tesis se realizó una demostración de una aplicación del método de sensado comprimido. Esta consiste en la obtención de fotografías digitales a partir de un dispositivo físico que utiliza únicamente un sensor como medio para realizar mediciones. Esta aplicación utiliza además 5 modelos basados en redes neuronales artificiales para realizar la reconstrucción de las fotografía finales a partir de las señales comprimidas obtenidas.

Con el trabajo se demostró que el desempeño del sensado comprimido, para el caso de imágenes digitales, depende en gran medida de la resolución de estas. A mayor resolución mejores resultados, lo cual fue notorio independientemente del modelo usado. En el caso de fotografías tomadas con el arreglo experimental en el laboratorio se pudo obtener una representación clara de una imagen de baja resolución realizando un muestreo de únicamente el 20 % de la señal, por lo que este porcentaje se considera como el mínimo para obtener un resultado decente. En el caso de imágenes digitales de alta resolución este mismo resultado se logró obteniendo únicamente un 1 % de la señal original.

En cuanto al desempeño de los modelos el peor lo obtuvo StanNet sin duda. En el caso del mejor no hay un ganador claro. Sin embargo basándose en los resultados para imágenes digitales, donde se pudo tener una comparación formal entre estos, el modelo LinearNet demostró ser mejor para imágenes de alta resolución mientras que DR2Net en general se desempeñó mejor con imágenes de baja resolución y sobre todo para porcentajes bajos de muestreo.

Finalmente se comentan dos puntos que podrían resultar interesantes para trabajos futuros. Estos dos parten del resultado sobre la resolución de las imágenes y el desempeño del CS. El primero es ver si en el caso específico de imágenes digitales, el CS puede mejorar los estándares de compresión, lo cual ayudaría a la transmisión de este tipo de señales. El segundo punto es probar qué tanto se pueden acercar los resultados de fotografías analógicas, aumentando la resolución, a los de fotografías digitales. Aquí el reto es ver cómo se puede modificar el arreglo experimental para volverlo más sensible a señales pequeñas o aumentar la potencia de estas, y por otro lado como reducir la magnitud del ruido que demostró tener un impacto considerable sobre la calidad y hace necesaria la obtención de varias mediciones del mismo valor para mitigar este efecto.

Apéndice A

Manipulación de muestreo físico

Lo que se busca aquí es justificar la forma en la que se modifica el muestreo obtenido del arreglo experimental ó muestreo físico para poder usarlo con los modelos. Lo primero que hay que recordar es que los modelos se entrenaron con el muestreo de diferentes imágenes digitales, los cuales se obtuvieron con las distintas matrices de sensado. Estas matrices siguen una distribución Bernoulli y sus entradas tienen 1's o -1's. Por otra parte tenemos los muestreos obtenidos con el arreglo experimental, en este caso lo que se hace es simular la matriz de sensado con el DMD. El comportamiento para las entradas con 1 es equivalente en los dos casos, digital y físico, sin embargo cuando la matriz de sensado tiene un -1, el comportamiento en el DMD es más bien como si tuviera un 0, ya que bloquea la luz de esa entrada. Por esta razón el muestreo físico y el muestreo digital no son equivalentes. El muestreo físico equivale a un muestreo digital donde a la matriz de muestreo correspondiente se le cambian los -1's por 0's. A continuación se hace el desarrollo para justificar el método utilizado para poder obtener un muestreo equivalente al digital.
Supongase una matriz de sensado $\Phi \in \mathbb{R}^{mxn}$ y una matriz $\Omega \in \mathbb{R}^{mxn}$ que se obtiene a partir de la matriz de sensado cambiando sus entradas con -1's por 0's. Denotamos los muestreos de una señal $\vec{x}$ con cada una de estas matrices como:

$$\vec{y_\Phi} = \Phi\vec{x} \tag{A.1}$$

$$\vec{y_\Omega} = \Omega\vec{x} \tag{A.2}$$

Lo que se tiene entonces es que $\vec{y_\Phi}$ representa el muestreo digital de la señal $\vec{x}$ mientras que $\vec{y_\Omega}$ representa el muestreo físico. Y lo que se quiere es obtener el primero en función del segundo.
Tomemos la entrada i de cada uno de los meustreos:

$$y_{\Phi i} = \sum_{j=1}^{n} \Phi_{i,j} x_j \tag{A.3}$$

$$y_{\Omega i} = \sum_{j=1}^{n} \Omega_{i,j} x_j \tag{A.4}$$

El siguiente paso es reacomodar y renombrar los elementos de las suma de tal forma que en las siguientes expresiones:

$$y_{\Phi i} = \sum_{j=1}^{n} \Phi'_{i,j} x'_j \tag{A.5}$$

$$y_{\Omega i} = \sum_{j=1}^{n} \Omega'_{i,j} x'_j \tag{A.6}$$

Se cumpla lo siguiente:

- Los elementos $\{\Phi'_i\}_{i=1}^{k}$ sean iguales a cero
- Los elementos $\{\Phi'_i\}_{i=k+1}^{n}$ sean iguales a 1
- Los elementos $\{\Omega'_i\}_{i=1}^{k}$ sean iguales a cero
- Los elementos $\{\Omega'_i\}_{i=k+1}^{n}$ sean iguales a 1

Tomando lo anterior en cuenta podemos expresar los muestreos como:

$$y_{\Phi i} = \sum_{j=k+1}^{n} x'_j \tag{A.7}$$

$$y_{\Omega i} = -(\sum_{j=1}^{k} x'_j) + (\sum_{j=k+1}^{n} x'_j) \tag{A.8}$$

Si se definen a y b de la siguiente manera:

$$a = \sum_{j=k+1}^{n} x'_j \tag{A.9}$$

$$b = \sum_{j=1}^{k} x'_j \tag{A.10}$$

Entonces se tiene que:

$$||\vec{x}||_{l_1} = a + b \tag{A.11}$$

Usando A.7 - A.11 se obtiene:

$$y_{\Phi i} = 2y_{\Omega i} - ||\vec{x}||_{l_1} \tag{A.12}$$

Que es la expresión que se usa para obtener el muestreo equivalente al muestreo digital a partir del obtenido físicamente. La única información extra que se necesita es $||\vec{x}||_{l_1}$ que corresponde a la intensidad total de la señal.

Apéndice B

PSNR fotografías reales

Aquí se presenta la información completa obtenida para el estudio de la relación que guarda la resolución con la calidad de la reconstrucción de una imagen. Esto corresponde al análisis que se lleva a cabo en la sección 6.1. En las tablas que se presentan a continuación se tiene el PSNR promedio obtenido para las 5 fotografías reales con cada modelo y para los 5 porcentajes de muestreo (1 %, 5 %, 10 %, 20 %, 40 %).

Foto	Modelo	Porcentaje de muestreo	PSNR en función de la dimensión (en pixeles) de la imagen												
			32	288	576	864	1152	1440	1728	2016	2304	2592	2880	3168	3456
Foto 1	StanNet	1%	12.69	20.35	22.28	23.42	23.94	24.47	24.83	25.01	25.21	25.36	25.50	25.61	25.65
		5%	15.58	25.69	28.94	30.69	31.34	32.27	32.91	33.11	33.44	33.65	33.91	34.09	33.94
		10%	17.31	27.03	30.77	32.69	33.39	34.48	35.20	35.40	35.79	36.02	36.31	36.51	36.34
		20%	17.14	27.76	31.53	33.53	34.24	35.38	36.18	36.35	36.74	37.00	37.32	37.55	37.29
		40%	17.53	28.15	32.00	34.06	34.72	35.83	36.66	36.80	37.21	37.46	37.77	38.00	37.74
	ReconNet	1%	13.64	19.48	21.18	21.93	22.37	22.78	23.07	23.20	23.36	23.49	23.60	23.70	23.73
		5%	14.94	26.87	30.39	32.28	33.15	34.23	35.09	35.33	35.74	36.04	36.36	36.64	36.42
		10%	17.79	28.87	32.55	34.51	35.19	36.37	37.30	37.48	37.93	38.22	38.56	38.83	38.46
		20%	18.94	30.53	34.45	36.17	36.51	37.75	38.75	38.86	39.36	39.63	40.06	40.36	39.93
		40%	20.65	32.79	36.50	38.09	38.15	39.54	40.76	40.85	41.50	41.84	42.43	42.85	42.27
	LinealNet	1%	14.64	21.74	24.19	25.36	26.05	26.72	27.15	27.41	27.65	27.88	28.06	28.23	28.27
		5%	15.05	26.86	31.13	33.58	34.52	36.08	37.29	37.63	38.30	38.71	39.26	39.71	39.29
		10%	17.50	28.74	33.17	35.55	36.20	37.80	39.17	39.40	40.14	40.56	41.25	41.76	41.14
		20%	19.25	30.84	35.32	37.37	37.60	39.37	41.05	41.22	42.17	42.64	43.61	44.29	43.51
		40%	20.27	33.33	37.48	39.29	39.15	41.23	43.42	43.70	45.05	45.59	47.11	47.98	47.19
	ReconNet2	1%	13.81	19.06	20.74	21.56	22.02	22.47	22.77	22.94	23.10	23.25	23.38	23.49	23.53
		5%	14.98	26.58	29.96	31.78	32.64	33.73	34.51	34.85	35.28	35.56	35.88	36.15	35.98
		10%	17.95	28.99	32.66	34.66	35.37	36.54	37.43	37.68	38.14	38.41	38.75	39.02	38.67
		20%	20.27	30.91	34.32	35.86	36.24	37.32	38.19	38.28	38.71	38.94	39.28	39.54	39.11
		40%	21.41	33.81	37.32	38.83	38.86	40.17	41.31	41.34	41.93	42.20	42.72	43.10	42.48
	DR2Net	1%	13.73	21.86	24.67	26.14	26.99	27.86	28.48	28.81	29.18	29.47	29.73	29.95	30.02
		5%	14.74	26.72	30.06	31.79	32.58	33.55	34.28	34.55	34.92	35.19	35.49	35.72	35.58
		10%	18.09	29.01	32.49	34.18	34.74	35.71	36.38	36.59	36.93	37.14	37.39	37.61	37.33
		20%	19.95	31.19	34.91	36.53	36.93	38.04	38.94	39.02	39.44	39.66	40.00	40.25	39.83
		40%	21.13	33.91	37.22	38.38	38.37	39.33	39.92	39.98	40.23	40.36	40.57	40.71	40.43
Foto 2	StanNet	1%	17.52	21.01	22.02	22.60	22.96	23.23	23.47	23.66	23.84	23.96	24.05	24.14	24.17
		5%	18.12	26.39	28.14	29.21	29.70	30.40	31.01	31.22	31.58	31.90	32.16	32.40	32.29
		10%	18.27	27.36	29.55	30.67	31.14	32.18	33.10	33.44	34.04	34.49	34.96	35.40	35.19
		20%	19.16	27.78	29.95	31.12	31.62	32.67	33.58	33.94	34.55	35.01	35.48	35.92	35.73
		40%	18.89	28.12	30.42	31.59	32.10	33.22	34.16	34.52	35.16	35.65	36.16	36.64	36.43
	ReconNet	1%	16.73	19.00	19.63	20.14	20.36	20.56	20.80	20.89	21.05	21.14	21.22	21.30	21.37
		5%	19.00	27.33	29.28	30.21	30.54	31.40	32.04	32.24	32.73	32.93	33.24	33.53	33.40
		10%	18.76	28.63	30.74	31.67	31.89	32.72	33.35	33.52	33.95	34.23	34.55	34.87	34.74
		20%	19.54	29.42	31.53	32.68	33.12	34.45	35.60	36.03	36.76	37.29	37.90	38.39	38.17
		40%	20.76	30.85	32.90	33.98	34.29	35.63	36.85	37.27	38.15	38.74	39.49	40.12	39.83
	LinealNet	1%	15.87	22.76	23.95	24.60	25.00	25.32	25.63	25.82	26.16	26.24	26.36	26.47	26.57
		5%	18.24	27.28	29.39	30.57	31.02	32.15	33.09	33.44	34.10	34.59	35.11	35.59	35.38
		10%	18.23	28.05	30.36	31.58	32.08	33.44	34.69	35.16	36.02	36.66	37.38	38.03	37.71
		20%	19.23	29.12	31.36	32.78	33.39	35.27	37.01	37.65	38.84	39.64	40.72	41.58	41.10
		40%	20.29	30.58	32.88	34.58	35.38	37.99	40.50	41.35	43.03	43.97	45.63	46.72	46.08
	ReconNet2	1%	18.16	18.50	19.19	19.65	19.88	20.05	20.29	20.38	20.57	20.64	20.72	20.81	20.87
		5%	19.33	27.64	29.57	30.54	30.88	31.63	32.26	32.39	32.72	32.97	33.22	33.44	33.33
		10%	19.22	29.03	31.19	32.14	32.41	33.27	33.97	34.13	34.54	34.85	35.18	35.48	35.31
		20%	20.56	30.25	32.18	33.04	33.19	34.11	34.85	35.00	35.47	35.79	36.16	36.50	36.31
		40%	22.24	31.81	33.71	34.64	34.80	36.06	37.13	37.46	38.20	38.72	39.35	39.89	39.60
	DR2Net	1%	18.15	23.42	24.98	25.75	26.17	26.60	27.05	27.23	27.54	27.71	27.88	28.03	28.08
		5%	18.82	27.47	29.41	30.35	30.74	31.61	32.25	32.47	32.93	33.14	33.43	33.68	33.60
		10%	18.60	28.62	30.80	31.80	32.22	33.18	34.01	34.29	34.79	35.14	35.53	35.84	35.71
		20%	20.10	30.08	32.23	33.34	33.69	34.98	36.01	36.34	36.99	37.43	37.95	38.38	38.15
		40%	21.07	31.54	33.76	35.12	35.60	37.43	38.85	39.27	39.98	40.32	40.73	40.96	40.77

Cuadro B.1

Foto	Modelo	Porcentaje de muestreo	PSNR en función de la dimensión (en pixeles) de la imagen												
			32	288	576	864	1152	1440	1728	2016	2304	2592	2880	3168	3456
Foto3	StanNet	1%	10.23	12.92	14.34	15.16	15.57	16.29	16.90	17.22	17.62	18.04	18.33	18.64	18.73
		5%	11.39	14.47	16.47	17.89	18.80	20.18	21.30	21.94	22.79	23.52	24.14	24.75	24.77
		10%	11.33	14.94	17.12	18.68	19.65	21.19	22.47	23.17	24.09	24.90	25.59	26.23	26.19
		20%	11.56	15.29	17.59	19.29	20.33	21.98	23.38	24.11	25.14	26.01	26.75	27.47	27.36
		40%	11.62	15.53	17.84	19.56	20.63	22.33	23.74	24.49	25.55	26.42	27.16	27.88	27.77
	ReconNet	1%	10.35	13.10	14.35	15.13	15.54	16.21	16.78	17.08	17.44	17.83	18.12	18.40	18.48
		5%	10.89	14.51	16.72	18.32	19.36	20.89	22.11	22.83	23.78	24.60	25.28	25.96	25.96
		10%	10.84	15.07	17.66	19.64	20.90	22.82	24.39	25.21	26.34	27.27	28.08	28.86	28.67
		20%	11.03	15.48	18.55	20.98	22.37	24.64	26.52	27.37	28.61	29.58	30.47	31.29	30.83
		40%	11.67	16.69	20.53	23.74	25.21	28.03	30.39	31.23	32.72	33.77	34.72	35.56	34.56
	LinealNet	1%	10.65	13.21	14.87	15.82	16.30	17.17	17.87	18.26	18.72	19.24	19.60	19.94	20.07
		5%	10.87	14.36	16.66	18.27	19.28	20.88	22.14	22.86	23.88	24.73	25.45	26.18	26.14
		10%	10.96	14.88	17.55	19.55	20.73	22.68	24.28	25.08	26.27	27.24	28.11	28.96	28.70
		20%	10.83	15.51	18.77	21.32	22.70	25.10	27.12	27.99	29.41	30.51	31.57	32.57	31.93
		40%	11.75	16.74	20.71	23.95	25.32	28.28	30.91	31.82	33.64	34.96	36.37	37.60	36.29
	ReconNet2	1%	10.44	12.88	14.16	14.90	15.27	15.93	16.48	16.79	17.13	17.52	17.79	18.06	18.14
		5%	11.09	14.64	16.89	18.46	19.49	20.99	22.18	22.88	23.83	24.62	25.29	25.94	25.96
		10%	10.92	15.21	17.89	19.90	21.13	23.03	24.58	25.40	26.51	27.43	28.22	28.99	28.81
		20%	11.49	16.30	19.51	22.06	23.46	25.81	27.70	28.56	29.81	30.75	31.59	32.35	31.80
		40%	12.76	17.57	21.52	24.82	26.23	29.19	31.65	32.50	34.00	35.02	35.95	36.77	35.64
	DR2Net	1%	10.32	13.18	14.95	16.02	16.63	17.58	18.37	18.81	19.41	19.99	20.42	20.87	20.99
		5%	10.97	14.59	16.84	18.44	19.47	20.99	22.18	22.89	23.82	24.61	25.28	25.93	25.94
		10%	10.82	15.21	17.90	19.90	21.12	23.01	24.53	25.34	26.40	27.27	28.02	28.72	28.51
		20%	10.88	15.95	19.28	21.92	23.34	25.72	27.64	28.50	29.77	30.71	31.55	32.31	31.78
		40%	12.33	17.30	21.36	24.67	26.05	28.93	31.26	32.05	33.41	34.30	35.08	35.73	34.77
Foto 4	StanNet	1%	22.66	25.08	25.30	25.38	25.38	25.47	25.52	25.52	25.53	25.55	25.57	25.59	25.56
		5%	30.02	33.02	33.51	33.80	33.70	34.19	34.56	34.61	34.79	34.93	35.12	35.25	35.10
		10%	31.30	34.38	34.98	35.34	35.22	35.88	36.44	36.48	36.80	36.98	37.24	37.46	37.24
		20%	31.70	35.12	35.76	36.18	36.03	36.79	37.44	37.50	37.88	38.12	38.45	38.71	38.46
		40%	31.85	35.51	36.17	36.56	36.40	37.17	37.80	37.88	38.26	38.51	38.85	39.11	38.87
	ReconNet	1%	21.12	22.90	23.02	23.04	23.01	23.07	23.11	23.05	23.07	23.08	23.07	23.09	23.09
		5%	31.33	34.61	35.34	35.64	35.41	35.99	36.47	36.45	36.67	36.79	36.98	37.14	36.87
		10%	31.48	35.77	36.51	36.88	36.59	37.30	37.85	37.78	38.07	38.23	38.49	38.73	38.40
		20%	32.55	36.19	36.91	37.40	37.24	38.23	39.11	39.21	39.71	40.00	40.44	40.77	40.47
		40%	34.24	37.07	37.79	38.39	38.28	39.54	40.72	40.92	41.65	42.07	42.73	43.23	42.81
	LinealNet	1%	23.12	28.04	28.29	28.35	28.31	28.49	28.56	28.51	28.57	28.60	28.61	28.66	28.60
		5%	31.22	35.36	36.30	36.83	36.55	37.57	38.41	38.45	38.94	39.23	39.68	40.06	39.60
		10%	31.40	35.78	36.70	37.27	37.08	38.30	39.47	39.62	40.32	40.75	41.41	41.94	41.42
		20%	32.50	36.26	37.11	37.86	37.80	39.49	41.14	41.46	42.49	43.06	44.10	44.84	44.21
		40%	34.61	37.11	37.96	39.07	39.19	41.53	43.94	44.44	45.95	46.60	48.24	49.19	48.56
	ReconNet2	1%	19.94	22.40	22.50	22.54	22.51	22.58	22.62	22.58	22.60	22.61	22.61	22.62	22.61
		5%	30.71	34.83	35.60	35.91	35.69	36.25	36.65	36.61	36.78	36.88	37.03	37.17	36.87
		10%	31.76	36.11	36.92	37.28	37.00	37.65	38.21	38.14	38.43	38.58	38.80	39.01	38.70
		20%	32.78	36.23	36.93	37.30	36.99	37.69	38.29	38.27	38.59	38.77	39.06	39.29	39.00
		40%	34.80	37.60	38.30	38.79	38.58	39.66	40.60	40.72	41.31	41.66	42.20	42.64	42.23
	DR2Net	1%	24.77	30.42	30.95	31.13	31.04	31.34	31.56	31.49	31.59	31.66	31.73	31.79	31.65
		5%	30.82	34.38	35.08	35.37	35.20	35.66	36.02	35.98	36.13	36.22	36.35	36.44	36.22
		10%	31.07	35.35	36.02	36.31	36.12	36.66	37.11	37.09	37.32	37.45	37.62	37.78	37.57
		20%	33.05	36.72	37.43	37.78	37.56	38.36	39.00	39.02	39.38	39.59	39.91	40.18	39.89
		40%	34.87	37.57	38.13	38.56	38.48	39.28	39.78	39.88	40.10	40.20	40.36	40.45	40.35

Cuadro B.2

Foto	Modelo	Porcentaje de muestreo	PSNR en función de la dimensión (en pixeles) de la imagen											
			32	288	576	864	1152	1440	1728	2016	2304	2592	2880	3168	3456
Foto 5	StanNet	1%	11.88	17.68	19.73	20.69	21.10	21.81	22.28	22.51	22.81	22.99	23.21	23.36	23.34
		5%	12.98	20.56	22.49	23.61	23.98	24.99	25.78	26.02	26.52	26.89	27.27	27.61	27.38
		10%	12.89	21.26	23.22	24.34	24.65	25.71	26.52	26.79	27.34	27.73	28.12	28.49	28.25
		20%	13.55	22.01	24.10	25.24	25.56	26.70	27.62	27.93	28.53	28.98	29.44	29.89	29.58
		40%	14.08	22.34	24.37	25.50	25.79	26.95	27.85	28.16	28.76	29.20	29.66	30.12	29.80
	ReconNet	1%	11.92	17.91	19.99	21.08	21.34	22.12	22.64	22.85	23.11	23.33	23.52	23.70	23.62
		5%	12.51	21.58	23.80	25.03	25.34	26.49	27.40	27.64	28.20	28.58	29.02	29.39	29.09
		10%	13.17	22.74	25.01	26.26	26.49	27.84	28.89	29.17	29.86	30.33	30.85	31.31	30.94
		20%	13.59	23.44	25.65	26.93	27.17	28.80	30.17	30.60	31.52	32.16	32.88	33.50	33.02
		40%	14.86	25.05	27.32	28.76	29.05	31.19	33.09	33.67	34.92	35.75	36.67	37.42	36.81
	LinealNet	1%	12.38	18.16	20.53	21.83	22.17	23.08	23.75	23.99	24.35	24.62	24.87	25.10	25.01
		5%	12.66	21.34	23.53	24.76	25.07	26.31	27.31	27.60	28.24	28.72	29.24	29.70	29.34
		10%	12.86	22.39	24.58	25.84	26.10	27.60	28.84	29.20	30.05	30.64	31.33	31.94	31.48
		20%	13.99	23.50	25.67	27.00	27.25	29.11	30.77	31.29	32.44	33.25	34.19	35.02	34.39
		40%	15.11	24.99	27.20	28.67	28.99	31.43	33.76	34.51	36.13	37.23	38.56	39.61	38.74
	ReconNet2	1%	12.18	18.20	20.39	21.53	21.75	22.57	23.14	23.33	23.64	23.85	24.06	24.24	24.18
		5%	12.76	21.70	23.99	25.24	25.55	26.66	27.54	27.74	28.26	28.63	29.01	29.37	29.06
		10%	13.34	22.91	25.18	26.45	26.66	27.97	28.96	29.20	29.84	30.27	30.74	31.15	30.80
		20%	14.32	24.36	26.58	27.84	28.02	29.57	30.80	31.14	31.93	32.47	33.04	33.56	33.12
		40%	15.11	25.78	28.10	29.55	29.80	31.87	33.60	34.08	35.16	35.89	36.67	37.34	36.76
	DR2Net	1%	12.11	18.11	20.48	21.86	22.17	23.14	23.81	24.06	24.44	24.72	24.99	25.23	25.10
		5%	12.78	21.76	23.94	25.19	25.51	26.59	27.44	27.64	28.14	28.50	28.87	29.20	28.93
		10%	13.47	22.96	25.18	26.42	26.63	27.92	28.90	29.13	29.76	30.17	30.63	31.02	30.67
		20%	13.96	24.25	26.52	27.80	27.98	29.50	30.69	31.01	31.75	32.26	32.79	33.27	32.86
		40%	15.32	25.77	28.02	29.45	29.70	31.73	33.43	33.92	34.97	35.68	36.41	37.02	36.49

Cuadro B.3

Apéndice C

Resultado señales con ruido

Aquí se presenta la información completa obtenida para señales con ruido para cada modelo. Está información consiste en la calidad promedio, medida como PSNR, de la reconstrucción de las imágenes en el conjunto CIFARtest muestreo con diferentes niveles de ruido para cada modelo y para cada porcentaje de muestreo. El ruido se genera a partir de distribuciones normales centradas en cero y lo que varía es su desviación estándar, la cual se denota como σ.

Modelo	Procentaje de Muestreo	desviación estándar							
		0	100	200	300	400	500	1000	2000
StanNet	1%	15.78	15.77	15.73	15.67	15.58	15.46	14.73	13.21
	5%	18.81	18.77	18.67	18.51	18.30	18.06	16.59	13.97
	10%	19.97	19.93	19.82	19.66	19.43	19.18	17.61	14.79
	20%	20.81	20.78	20.68	20.53	20.33	20.09	18.61	15.81
	40%	21.13	21.10	21.03	20.92	20.77	20.59	19.40	16.85
ReconNet	1%	15.76	15.74	15.71	15.65	15.57	15.48	14.81	13.17
	5%	19.36	19.32	19.22	19.04	18.81	18.52	16.71	13.29
	10%	21.35	21.29	21.10	20.81	20.41	19.94	17.25	13.03
	20%	24.08	23.94	23.52	22.86	22.05	21.15	16.90	11.73
	40%	27.79	27.29	26.06	24.51	22.94	21.45	15.97	10.38
LinealNet	1%	15.79	15.77	15.73	15.67	15.58	15.48	14.71	12.71
	5%	19.14	19.10	18.99	18.80	18.57	18.30	16.58	13.08
	10%	21.16	21.08	20.85	20.52	20.11	19.63	17.04	12.69
	20%	23.86	23.66	23.12	22.38	21.55	20.69	16.81	11.60
	40%	28.07	27.25	25.56	23.77	22.13	20.65	15.38	9.57
ReconNet2	1%	15.86	15.85	15.81	15.74	15.66	15.57	14.84	13.03
	5%	19.46	19.42	19.31	19.13	18.90	18.61	16.78	13.46
	10%	21.65	21.58	21.37	21.04	20.61	20.09	17.19	12.69
	20%	24.76	24.58	24.07	23.31	22.43	21.49	17.45	13.05
	40%	29.16	28.46	26.72	24.64	22.68	20.97	15.35	10.08
DR2Net	1%	15.90	15.89	15.86	15.79	15.71	15.61	14.87	13.07
	5%	19.47	19.43	19.33	19.14	18.91	18.62	16.80	13.36
	10%	21.71	21.64	21.41	21.07	20.63	20.11	17.35	13.02
	20%	24.65	24.48	23.98	23.21	22.27	21.26	16.72	11.42
	40%	29.16	28.43	26.69	24.67	22.79	21.12	15.42	9.55

Cuadro C.1

Bibliografía

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Emmanuel J. Candès y Michael B. Wakin. «An Introduction to Compressive Sampling». En: *IEEE Signal Processing Magazine* 25 (2008), págs. 21-30. DOI: 10.1109/MSP.2007.914731.

[3] François Chollet y col. *Keras*. https://keras.io. 2015.

[4] Simon Haykin. *Neural Networks and Learning Machines*. Pearson, 2008.

[5] D. O. Hebb. *The organization of behavior; a neuropsychological theory*. Wiley, 1949.

[6] Aaron Courville Ian Goodfellow Yoshua Bengio. *Deep Learning*. MIT press, 2016.

[7] Ori Katz, Yaron Bromberg y Yaron Silberberg. «Compressive ghost imaging». En: *Applied Physics Letters* (sep. de 2009), pág. 95.

[8] Diederik P. Kingma y Jimmy Ba. «Adam: A Method for Stochastic Optimization». En: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)* (dic. de 2014).

[9] Donald Knuth. *Knuth: Computers and Typesetting*. URL: http://www-cs-faculty.stanford.edu/~uno/abcde.html. (accessed: 01.09.2016).

[10] A. Krizhevsky y G. E. Hinton. «Learning Multiple Layers of Features from Tiny Images». En: *Technical report, University of Toronto* (2009).

[11] Kuldeep Kulkarni y col. «ReconNet: Non-Iterative Reconstruction of Images from Compressively Sensed Random Measurements». En: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, págs. 449-458.

[12] Tsung-Yi Lin y col. «Microsoft COCO: Common Objects in Context». En: European conference on computer vision. Springer, Cham, 2014, págs. 740-755.

[13] W.S. McCulloch y W. Pitts. «A logical calculus of the ideas immanent in nervous activity». En: *Bulletin of Mathematical Biophysics 5* (1949), págs. 115-133.

[14] Phan Minh Nguyen. «A Learning Approach to Compressed Sensing». En: (2017).

[15] Michael Nielsen. *Neural Networks and Deep Learning*. 2015. URL: `http://neuralnetworksanddeeplearning.com`.

[16] A. V. Oppenheim y R. W. Schafer. *Discrete-time signal processing*. Prentice Hall, 1989.

[17] A.J.M.M. Weijters P.J. Braspenning F. Thuijsman. *Artificial Neural Networks An Introduction to ANN Theory and Practice*. Springer, 1991.

[18] F. Rosenblatt. «The perceptron: A probabilistic model for information storage and organization in the brain.» En: *Psychological Review* 65(6) (1958), págs. 386-408.

[19] J.J. SPILKER Jr. «GPS signal structure and performance characteristics». En: *Navigation* 25 (1978), págs. 121-146.

[20] Hantao Yao y col. «DR2-Net: Deep Residual Reconstruction Network for Image Compressive Sensing». En: *Neurocomputing* 359 (sep. de 2019), págs. 483-493.