Evaluación de técnicas para mejorar la inicialización de parámetros en redes neuronales convolucionales.

Daniel Fernando Gómez Guzmán

Ingeniería mecatrónica

Olac Fuentes
Diego Aguirre
Gloria Diaz Cabrera

INSTITUTO TECNOLÓGICO METROPOLITANO
31/07/2018

RESUMEN

Este documento corresponde al reporte de resultados del trabajo realizado como pasantía de investigación en la Universidad de Texas el Paso (UTEP) en el marco del proyecto: “Aplicaciones de visión por computador: programa conjunto de formación en investigación itm – utep”.

El objeto del trabajo fue evaluar el desempeño de un esquema de inicialización de parámetros, propuesto por los investigadores de UTEP Diego Aguirre y Olac Fuentes, y compararlo con esquemas clásicos. En el primer conjunto de experimentos se evalúa el esquema variando sus hiperparámetros para hallar valores óptimos y en el segundo conjunto de experimentos se compara su desempeño con esquemas de inicialización clásicos. Los dos conjuntos de experimentos se realizan en dos redes convolucionales preentrenadas y en una red neuronal convolucional. Además, se evalúa la red neuronal, con el nuevo esquema, preinicializando sus capas convolucionales con filtros de Gabor y con matrices ortogonales para comparar su desempeño. En todos los experimentos realizados el esquema de inicialización propuesto por los investigadores de la UTEP converge más rápido y generaliza mejor que los esquemas clásicos.

*Palabras clave***:** redes neuronales, inicialización alternativa, Gram-Schmidt en redes neuronales,, filtros de Gabor en redes.

RECONOCIMIENTOS

Estoy profundamente agradecido con mis padres y familiares por brindarme su apoyo incondicional y sus valiosos consejos en el transcurso del pregrado. Además agradezco a los doctores Carlos Andrés Madrigal González, Olac Fuentes y Diego Aguirre por sus enseñanzas en el área de redes neuronales, su paciencia y por darme acceso a computadores con especificaciones adecuadas para entrenar redes neuronales convolucionales.

Este informe fue financiado en parte por una subvención del Departamento de Estado, Embajada de los EEUU en Bogotá, Colombia, Sección de Asunto Públicos y administrado por Partners of the Americas. Las opiniones, hallazgos y las conclusiones aquí mencionadas son las del autor(es) y no necesariamente reflejan las del Departamento de Estado de EEUU o Partners of the Americas.

ACRÓNIMOS

LSUV Varianza de unidad secuencial de capa

Arquit.: Arquitectura

BD base de datos

desv.: desviación estándar

TABLA DE CONTENIDO

1. INTRODUCCIÓN

El aprendizaje profundo ha revolucionado la visión por computador desde el 2010 (Chollet, 2018). El aprendizaje profundo ha logrado resultados increíbles en aplicaciones de visión por computador y ha estado muy cerca de igualar el desempeño de los humanos en tareas como clasificación de imágenes y conducción autónoma (Chollet, 2018).

Las arquitecturas empleadas en aprendizaje profundo usualmente reciben el nombre de redes neuronales profundas o redes neuronales convolucionales. Una red neuronal convolucional esta compuesta de capas convolucionales y capas densas. Las capas convolucionales extraen características de la imagen utilizando pesos sinápticos. Una capa densa se conecta con otras capas a través de pesos sinápticos. Usualmente los pesos sinápticos son llamados parámetros de la red.

Las redes profundas son entrenadas en lugar de explícitamente programadas (Chollet, 2018). El entrenamiento de las redes profundas consiste en presentar a la red muchas imágenes, las cuales son representativas de una tarea o fenómeno físico, para que la red encuentre la función matemática de la tarea o fenómeno, la función esta oculta en la base de datos (Chollet, 2018). Las redes profundas entrenadas pueden predecir con alta exactitud las imágenes de diversos fenómenos físicos y por esa razón se han utilizado para automatizar aplicaciones de visión por computador. El problema con las redes profundas es que pueden contener millones de parámetros y entrenar la red toma demasiado tiempo (Chollet, 2018).

Antes de comenzar el entrenamiento de la red se deben asignar valores aleatorios a los parámetros, por ende la salida de la red estará alejada de la salida deseada (Chollet, 2018). Asignar valores iniciales a los parámetros se denomina inicialización.

La inicialización de parámetros es un factor determinante en el tiempo de entrenamiento y en la calidad de las predicciones de la red. Una inicialización inadecuada en arquitecturas profundas puede implicar que la red no encuentre una función adecuada para predecir la tarea o fenómeno físico (He et al, 2015).

Objetivo general

Evaluar el desempeño de un método de inicialización de parámetros para el entrenamiento de redes neuronales, propuesto por investigadores de la UTEP, frente a esquemas de inicialización clásicos.

Objetivos específicos.

1. Realizar una revisión de literatura de trabajos previos de inicialización de parámetros para seleccionar un conjunto de métodos clásicos como base de comparación.

2. Implementar los métodos seleccionados para inicializar los parámetros de redes convolucionales.

3. Identificar los parámetros óptimos para el esquema de inicialización de parámetros propuesto por los investigadores de la UTEP.

4. Comparar el desempeño de los métodos de inicialización mediante funciones de costo y de testeo.

Este documento se encuentra organizado de la siguiente manera, en el marco teórico se introducen los conceptos de redes neuronales convolucionales, se presentan los esquemas clásicos de inicialización, el nuevo esquema y conceptos requeridos para comprender este informe. En el capítulo de metodología se describen las redes neuronales preentrenadas que fueron usadas en los experimentos, la tarea de clasificación usada como prueba, y el diseño experimental seguido en el trabajo. En el capítulo 4 se presentan y analizan los resultados experimentales a través de gráficas y tablas. Finalmente, en la sección 5 se encuentran las conclusiones experimentales y se dan a conocer ideas que pueden mejorar este trabajo.

2. MARCO TEÓRICO

2.1. Fundamentos de redes neuronales convolucionales

Las redes neuronales profundas son aproximadores universales de funciones, para cualquier función hay garantía que existe una red neuronal profunda que puede modelarla (Nielssen, 2015). El fin de una red profunda es que la función modelada por la red pueda predecir un fenómeno físico, con un porcentaje alto de exactitud.

Las capas convolucionales están compuestas de un numero de filtros, donde cada filtro extrae características de la imagen que entra a la capa. La salida de la capa convolucional es la concatenación de las características extraídas por sus filtros (Nielssen, 2015). Los filtros convolucionales extraen características utilizando pesos sinápticos. Las capas densas están compuestas por un numero de neuronas, donde cada neurona se conecta a la capa anterior con pesos sinápticos. Usualmente los pesos sinápticos son llamados parámetros de la red. La salida de una capa densa es una representación unidimensional de las características que entraron a la capa.

Una red profunda requiere una base de datos numerosa, donde las subcarpetas indican la salida deseada en la red. Para medir la distancia entre la salida deseada y la salida real de la red se utiliza una función de costo, es decir que la función de costo indica que tan equivocada esta la red (Chollet, 2018). El valor de la función de costo se realimenta a la red, esta señal la utiliza un algoritmo de optimización para modificar los parámetros de la red de tal forma que la salida sea mas cercana a la salida deseada. Ajustar progresivamente los parámetros de la red posibilita el aprendizaje paulatino de la red (Chollet, 2018).

Los parámetros de una red, es decir sus pesos sinápticos, ajustan sus valores a medida que la red entrena. Cuando finaliza el entrenamiento los valores de los pesos sinápticos son los coeficientes de la función que modela la red (Nielssen, 2015).

El algoritmo de optimización mas básico es el descenso del gradiente. Este algoritmo ajusta cada parámetro alejándose del gradiente de la función de costo respecto a dicho parámetro (Nielssen, 2015). La velocidad con la cual los parámetros se alejan del gradiente de la función de costo es llamada tasa de aprendizaje, la cual se denota con la letra griega alpha.

$$\Delta w_{ij} = -\alpha \cdot \frac{\partial E}{\partial w_{ij}}$$

Cuando la red entrena con todas las imagenes de la base de datos se dice que ha completado una época.

La inicialización de parámetros es un factor determinante en el tiempo de entrenamiento y en la calidad de las predicciones de la red. Una inicialización inadecuada en arquitecturas profundas puede implicar que la red no encuentre una función adecuada para predecir la tarea o fenómeno físico (He et al, 2015).

2.2. Esquemas de inicialización de parámetros en redes neuronales

Los esquemas Xavier, He y LSUV son muy utilizados en redes neuronales profundas y por esa razón muchas librerías como Keras, Tensorflow y Theano incluyen funciones que realizan estas inicializaciones.

El esquema Xavier inicializa los parámetros de cada capa de la red utilizando una distribución normal con media igual a cero y desviación estándar $\sqrt{2/Nin+Nout}$ (Glorot & Bengio, 2010), donde Nin es el número de entradas de la capa y Nout es el número de salidas de la capa.

El esquema He inicializa los parámetros de la red utilizando una distribución normal con media igual a cero y desviación estándar $\sqrt{2/Nin}$ (He et al, 2015), donde Nin es el número de entradas de la capa.

El esquema Xavier, a diferencia del esquema He, no es adecuado para capas donde se utilice la función de activación Relu (He et al, 2015).

En el esquema LSUV los pesos sinápticos son preinicializados con matrices ortonormales, posteriormente se obliga a que la varianza de cada capa sea igual a 1 utilizando un subconjunto de la base de datos (Mishkin & Matas, 2015). El esquema LSUV trabaja correctamente con diversas funciones de activación (Mishkin & Matas, 2015).

El esquema propuesto por Diego Aguirre y Olac Fuentes sera publicado en un futuro. En dicho documento se explicara detalladamente todos los pasos del esquema. El procedimiento general del esquema es el siguiente:

1. Inicializar los parámetros con matrices ortogonales de Gram-Schmidt. Con el método Gram-Schmidt a partir de un vector inicial se puede obtener un conjunto de vectores mutuamente ortogonales (Arfken, 1985). Los filtros de Gabor son filtros que extraen características de textura de la imagen.

2. Obligar a las capas a tener una desviación estándar predeterminada para un subconjunto de la base de datos llamado init_batch.

3. El nuevo esquema introduce el hiperparámetro non_zero_frac para controlar el porcentaje de salidas no activas de cada neurona Relu.

4. Para inicializar la capa final el esquema halla los parámetros multiplicando la salida deseada por la pseudoinversa Moore-Penrose de la penúltima capa, para el subconjunto init_batch.

3. METODOLOGÍA

3.1. Tarea de clasificación

Los experimentos se realizaron con la base de datos libre Flowers. Flowers se encuentra disponible en la pagina oficial de Tensorflow Slim. Esta base de datos libre esta conformada por 3670 imagenes divididas en 5 clases: 633 margaritas, 898 dientes de león, 641 rosas, 699 girasoles, 799 tulipanes. Las imagenes son RGB y presentan diferentes tamaños así que se tienen que redimensionar a un tamaño fijo. La tarea es clasificar correctamente estas clases de flores.

Para realizar los experimentos con las redes preentrenadas se redimensiona la base de datos a 299*299*3, para Inception, y a 224*224*3, para VGG19. La base de datos se dividió de la siguiente manera: El conjunto de evaluación se obtiene extrayendo 60 imágenes de cada una de las 5 clases, es decir que son 300 imágenes de evaluación, y el entrenamiento se realiza con las 3370 imágenes restantes.

Para realizar los experimentos con la red neuronal convolucional se redimensiona la base de datos a 90*90*3. La base de datos se dividió de la siguiente manera: El conjunto de evaluación se obtiene extrayendo el 20% de las imágenes de cada una de las 5 clases, es decir que son 840 imágenes de evaluación, y el entrenamiento se realiza con las 2830 imágenes restantes.

3.2. Arquitecturas empleadas en la evaluación

InceptionV3 y VGG19 se muestran en los apéndice B y C respectivamente.

InceptionV3 tiene más capas que VGG19, es decir que es más profunda. A pesar de que es más profunda tiene menos parámetros que vgg19 y por ello implica menos cálculos y un menor costo computacional (Szegedy et al, 2015). Inception requiere pocos parámetros debido a que está compuesta principalmente por módulos de incepción dispuestos en serie.

En Inception cada modulo de incepción está compuesto por diversos filtros convolucionales (1*1, 3*3, 5*5) y capas de pooling (maxPooling, avgPooling) dispuestos en serie y en paralelo. Los mapas de características resultantes de cada línea paralela se concatenan para formar un nuevo mapa de características que será la entrada de otro modulo de incepción o de una capa densa. Para concatenar varios mapas de características se tiene que cumplir la condición de que todos los mapas presenten la misma altura y ancho, no importa si la

profundidad de cada mapa es diferente ya que la profundidad del mapa resultante de la concatenación es la suma de las profundidades de los mapas que componen la concatenación. Inception utiliza un clasificador auxiliar para evitar que las primeras capas de la red aprendan muy lentamente (Szegedy et al, 2015). Todos los detalles de la arquitectura InceptionV3 se encuentran en (Szegedy et al, 2015).

La red neuronal empleada en este informe se muestra en el apéndice A. Esta red hace uso de un modulo de incepción.

VGG19 utiliza filtros convolucionales de tamaño 3*3 con paso igual a 1. Todos los detalles de la arquitectura VGG se encuentran en (Simonyan & Zisserman, 2014).

3.3. Diseño experimental

Los experimentos se llevaron a cabo en un portátil EVGA SC15 con las siguientes especificaciones: GPU GEFORCE GTX 1060 6GB, 16GB RAM, procesador intelcore i7-7700HQ. Toda la programación se realizo con Python, Tensorflow y Keras.

Todos los experimentos emplean la función de costo entropia cruzada, descenso del gradiente clásico, tasa de aprendizaje 0.001 para 50 épocas.

Se evalúa la exactitud y costo de las redes preentrenadas a medida que se varían los hiperparámetros init_batch, non_zero_frac y desviación estándar en su capa densa. Esto se hace con el objetivo de inicializar adecuadamente las capas densas de la red convolucional y de las redes convolucionales preentrenadas.

Cuando se inicializa adecuadamente las capas densas de la red convolucional se procede a evaluar la exactitud y costo de la red a medida que se varían los hiperparámetros non_zero_frac y desviación estándar en sus capas convolucionales, las cuales son preinicializadas con Gram-Schmidt y filtros de Gabor. Esto se hace con el objetivo de inicializar adecuadamente las capas convolucionales de la red. En las capas convolucionales se utilizan 2 técnicas de preinicialización para escoger la técnica con exactitud mas alta.

Para comparar los esquemas se grafican las funciones de exactitud y costo de la red convolucional y las redes convolucionales preentrenadas para cada inicialización.

3.3.1. Métrica de evaluación

La calidad de las predicciones se mide con el porcentaje de imágenes clasificadas correctamente, del conjunto de evaluación.

3.3.2. Selección de hiperparámetros del modelo propuesto por la UTEP

El valor adecuado para init_batch es el que obtiene la mayor exactitud final para Inception y VGG.

Un valor adecuado para non_zero_frac, en las capas densas, se encuentra en un intervalo donde se obtienen exactitudes altas con VGG e Inception, para diferentes desviaciones estándar en las capas densas.

Un valor adecuado para la desviación estándar, en las capas densas, se encuentra en un intervalo donde se obtienen exactitudes altas con VGG e Inception, para diferentes non_zero_frac en las capas densas.

Un valor adecuado para non_zero_frac, en las capas convolucionales, se encuentra en un intervalo donde se obtienen exactitudes altas con la red, para una desviación estándar arbitraria en las capas convolucionales.

Un valor adecuado para la desviación estándar, en las capas convolucionales, se encuentra en un intervalo donde se obtienen exactitudes altas con la red, para un non_zero_frac arbitrario en las capas convolucionales.

3.3.3. Comparación con modelos de inicialización clásicos

La primera versión modificada de LSUV obtiene las matrices ortogonales con la técnica Gram-Schmidt en vez de utilizar la técnica nombrada en la publicación original. La segunda versión modificada de LSUV preinicializa los parámetros con el esquema Xavier en vez de utilizar matrices ortogonales. En ambas versiones de LSUV se obliga a que la varianza de cada capa sea igual a 1 utilizando un subconjunto de la base de datos.

Se inicializan las capas densas, de las redes preentrenadas, con el nuevo esquema para obtener la función de costo y la función de exactitud. Se procede a inicializar las capas densas, de las redes preentrenadas, con los esquemas Xavier, las 2 versiones modificadas de LSUV y He para obtener las funciones de costo y exactitud. Se grafican todas las funciones de exactitud en un mismo plano de coordenadas. Se grafican todas las funciones de costo en un mismo plano de coordenadas.

Se inicializa la red con el nuevo esquema para obtener la función de costo y la función de exactitud. Se procede a inicializar la red con los esquemas Xavier, las 2 versiones modificadas de LSUV y He para obtener las funciones de costo y exactitud. Se grafican todas las funciones de exactitud en un mismo plano de coordenadas. Se grafican todas las funciones de costo en un mismo plano de coordenadas.

4. RESULTADOS Y DISCUSIÓN

Las siguientes tablas muestran los efectos de modificar el hiperparámetro init_batch en las redes preentrenadas VGG19 e InceptionV3.

Tabla 1. Modificación de init_batch en VGG19

1 capa oculta de 1000 neuronas semilla de entrenamiento:3 semilla de orden de la BD:3 train_batch: 128 arquitectura: vgg19 desviación estándar:0.3 non_zero_frac: 0.5						
Init batch	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
32	0,5633	0,8866	0,7341	0,3095	0,8966	0,1498
64	0,7133	0,89	0,6324	0,2516	0,91	0,1043
128	0,72	0,9033	0,5436	0,1591	0,9133	0,0457
256	0,7233	0,9266	0,9083	0,0815	0,9233	0,0159

tabla 2. Modificación de init_batch en InceptionV3

1 capa oculta de 1000 neuronas semilla de entrenamiento:3 semilla de orden de la BD:3 train_batch: 128 arquitectura: inceptionV3 desviación estándar: 0.3 non_zero_frac: 0.5						
Init batch	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
32	0,64	0,79	0,932	0,7068	0,85	0,4578
64	0,72	0,8233	0,7754	0,602	0,8533	0,3967
128	0,81	0,8566	0,6491	0,5178	0,8766	0,3383
256	0,8233	0,87	0,5589	0,4137	0,9	0,2449

Se observa que la exactitud final y el costo final mejoran a medida que aumenta el valor de init_batch. Esta relación es consecuencia de que un valor más alto de init_batch implica que la red inicie su entrenamiento con un mayor número de nociones del fenómeno físico analizado, puesto que la red inicia clasificando correctamente un numero de imágenes aproximado a init_batch. Un mayor init_batch es mejor que un init_batch pequeño solo si el grupo de imágenes del init_batch grande es suficientemente representativo del fenómeno. Un init_batch alto implica que el algoritmo de la pseudoinversa Moore-Penrose tenga en cuenta un mayor número de mapas característicos de imágenes para la inicialización de la capa final.

Las siguientes tablas exponen los resultados de modificar el hiperparámetro non_zero_frac en las capas densas de las redes preentrenadas. La modificación de non_zero_frac se realizó para desviación estándar:0.3, desviación estándar:0.7, desviación estándar:0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1 en InceptionV3 y VGG19.

Tabla 3. Modificación de non_zero_frac en VGG19 para desv:0.3

1 capa oculta de 1000 neuronas semilla de entrenamiento:3 semilla de orden de la BD:3 train_batch: 128 arquitectura: vgg19 desviación estándar: 0.3 init batch:256						
non_zero_frac	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,51	0,8566	1,2023	0,1241	0,8966	0,0177
0,2	0,6366	0,89	0,6684	0,0912	0,9133	0,0149
0,3	0,66	0,9033	0,5732	0,0812	0,9033	0,0142
0,4	0,69	0,9033	0,8021	0,0862	0,9066	0,0163
0,5	0,7233	0,92	0,9083	0,0816	0,92	0,0159
0,6	0,74	0,92	1,9838	0,102	0,92	0,0207
0,7	0,72	0,92	1,7388	0,0849	0,9133	0,0162
0,8	0,7233	0,9033	3,5118	0,1016	0,91	0,0189
0,9	0,6966	0,91	4,4629	0,1076	0,8966	0,0191
1	0,6999	0,89	13	0,11	0,9066	0,0125

Tabla 4. Modificación de non_zero_frac en VGG19 para desv:0.7

1 capa oculta de 1000 neuronas semilla de entrenamiento:3 semilla de orden de la BD:3 train_batch: 128 arquitectura: vgg19 desviación estándar: 0.7 init batch:256						
non_zero_frac	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,51	0,8266	1,6635	0,4532	0,8666	0,1066
0,2	0,6366	0,88	0,8792	0,3016	0,8966	0,079
0,3	0,66	0,88	0,698	0,2558	0,9	0,0682
0,4	0,69	0,8966	0,6262	0,2244	0,8933	0,0649
0,5	0,7233	0,9033	0,5789	0,1998	0,91	0,0585
0,6	0,74	0,8966	0,5552	0,1828	0,91	0,0541
0,7	0,72	0,9	0,5494	0,171	0,8966	0,0522
0,8	0,7233	0,8866	0,747	0,1621	0,9	0,0491
0,9	0,6966	0,89	1,39	0,1599	0,9	0,0444
1	0,6999	0,8	4,53	0,36	0,8	0,0566

Tabla 5. Modificación de non_zero_frac en VGG19 para desv:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1

1 capa oculta de 1000 neuronas semillas de entrenamiento:1,3,5,7,11,13,17,21,28,33 semilla de orden de la BD:3 train_batch: 128 arquitectura: vgg19 desviación estándar:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1 init batch:256						
non_zero_frac	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,4223	0,8231	8,3967	0,3356	0,861	0,0699
0,2	0,6189	0,8732	1,7268	0,2256	0,8955	0,0606
0,3	0,6876	0,8878	1,7484	0,1939	0,9052	0,0538
0,4	0,7143	0,8933	1,7124	0,1772	0,9102	0,0497
0,5	0,7256	0,8979	2,0573	0,1652	0,9118	0,0461
0,6	0,739	0,899	2,5586	0,1561	0,9122	0,0432
0,7	0,737	0,9004	3,2832	0,1486	0,914	0,0402
0,8	0,7196	0,9015	4,0275	0,1425	0,9116	0,0379
0,9	0,7016	0,9044	5,9324	0,1378	0,9112	0,0352
1	0,6693	0,893	12,72	0,2342	0,8996	0,0385

Tabla 6. Modificación de non_zero_frac en InceptionV3 para desv:0.3

1 capa oculta de 1000 neuronas semilla de entrenamiento:3 semilla de orden de la BD:3 train_batch: 128 arquitectura:inceptionV3 desviación estándar: 0.3 init batch:256						
non_zero _frac	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,7466	0,7833	0,8245	0,6947	0,8166	0,479
0,2	0,7933	0,82	0,6565	0,5284	0,8533	0,3394
0,3	0,8233	0,85	0,6045	0,4712	0,8766	0,2919
0,4	0,8266	0,8666	0,5754	0,4381	0,8866	0,264
0,5	0,8233	0,87	0,5589	0,4137	0,9	0,2449
0,6	0,8333	0,8666	0,5465	0,3943	0,9033	0,2296
0,7	0,83	0,87	0,5413	0,3794	0,8966	0,2181
0,8	0,83	0,87	0,5479	0,3699	0,8933	0,2073
0,9	0,8066	0,8766	0,5573	0,3603	0,8866	0,1959
1	0,7833	0,8633	0,5772	0,3563	0,8933	0,1914

Tabla 7. Modificación de non_zero_frac en InceptionV3 para desv:0.7

1 capa oculta de 1000 neuronas semilla de entrenamiento:3 semilla de orden de la BD:3 train_batch: 128 arquitectura:inceptionV3 desviación estándar: 0.7 init batch:256						
non_zero _frac	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,7466	0,7733	0,8253	0,7081	0,82	0,5007
0,2	0,7933	0,82	0,6587	0,5558	0,8566	0,3787
0,3	0,8233	0,84	0,6083	0,5047	0,8766	0,3372
0,4	0,8266	0,8566	0,5807	0,4796	0,8833	0,3173
0,5	0,8233	0,86	0,5658	0,4621	0,8933	0,3049
0,6	0,8333	0,86	0,5551	0,4505	0,8833	0,2959
0,7	0,83	0,86	0,5519	0,4426	0,89	0,291
0,8	0,83	0,85	0,5608	0,4422	0,8933	0,2897
0,9	0,8066	0,86	0,5734	0,4436	0,89	0,2846
1	0,7833	0,83	0,5988	0,4531	0,89	0,2865

Tabla 8. Modificación de non_zero_frac en InceptionV3 para desv:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1

1 capa oculta de 1000 neuronas semillas de entrenamiento:1,3,5,7,11,13,17,21,28,33 semilla de orden de la BD:3 train_batch: 128 arquitectura: inceptionV3 desviación estándar:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1 init batch:256						
non_zero _frac	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,711	0,7657	0,8158	0,666	0,8196	0,4413
0,2	0,785	0,8243	0,6419	0,51	0,8546	0,3305
0,3	0,807	0,8385	0,5893	0,4574	0,8642	0,294
0,4	0,8116	0,8419	0,5631	0,4283	0,8692	0,274
0,5	0,8136	0,847	0,5485	0,4101	0,8712	0,2619
0,6	0,8146	0,8489	0,5384	0,3959	0,8745	0,2518
0,7	0,8193	0,8503	0,5328	0,3854	0,8766	0,2437
0,8	0,8183	0,85	0,5312	0,3779	0,8765	0,2376
0,9	0,8133	0,8524	0,5448	0,3718	0,8775	0,232
1	0,8063	0,8512	0,6135	0,3726	0,8776	0,2307

Un valor adecuado para non_zero_frac, en estas redes preentrenadas, se encuentra en el intervalo 0.5 a 0.8. El intervalo anterior es más evidente en VGG que en la red Inception.

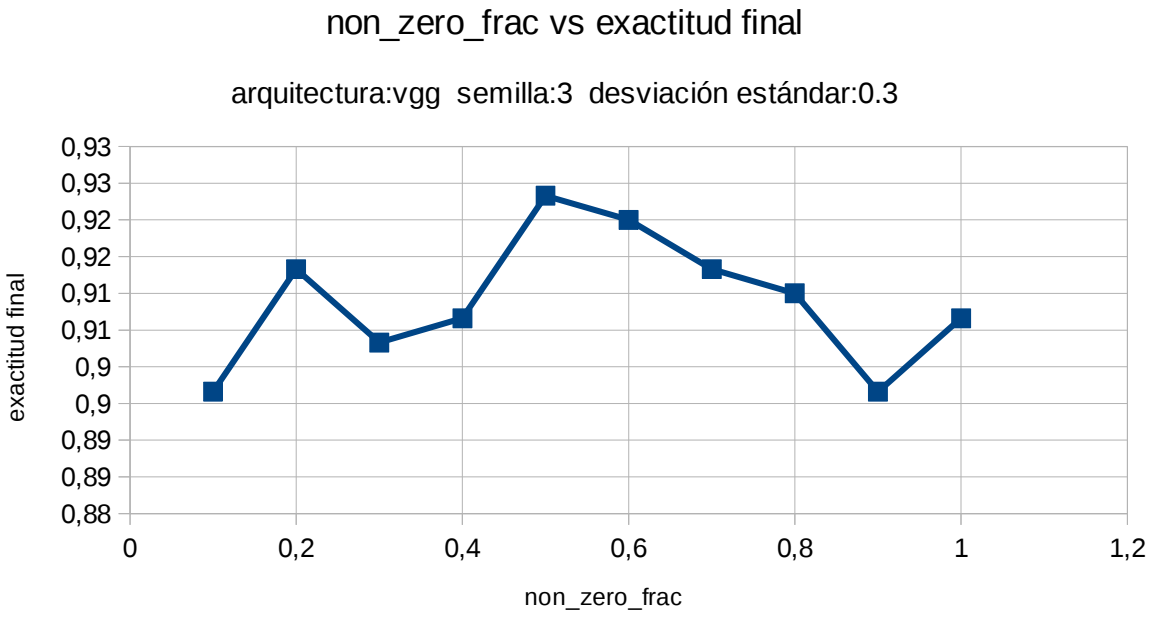


Figura 1. non_zero_frac vs exactitud final en VGG para desv.:0.3

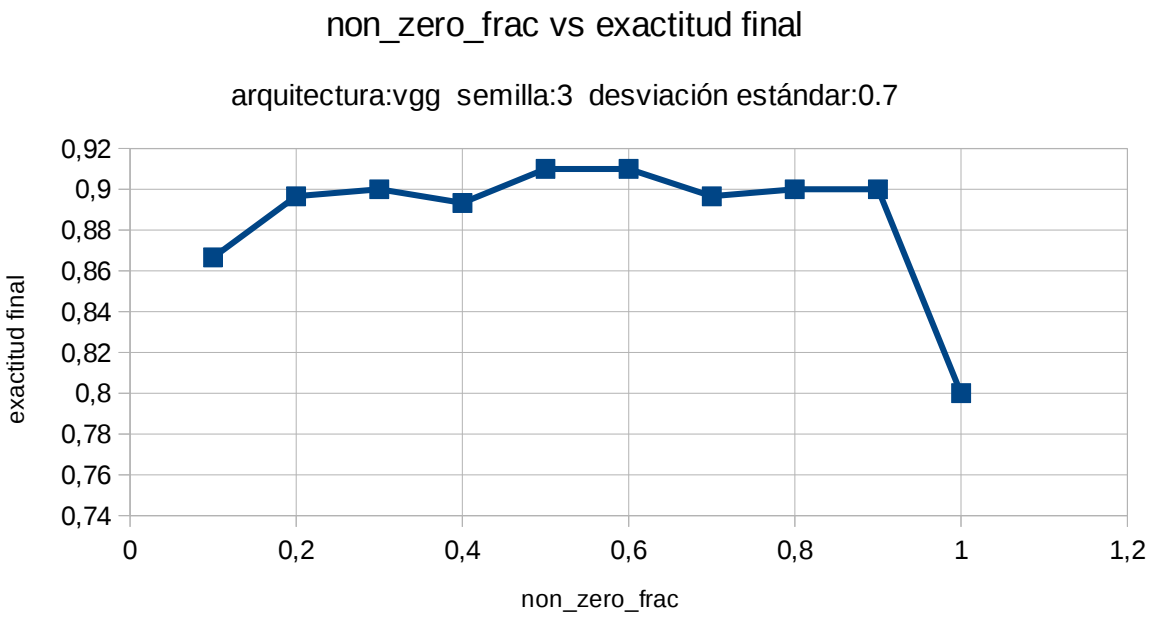


Figura 2. non_zero_frac vs exactitud final en VGG para desv.:0.7

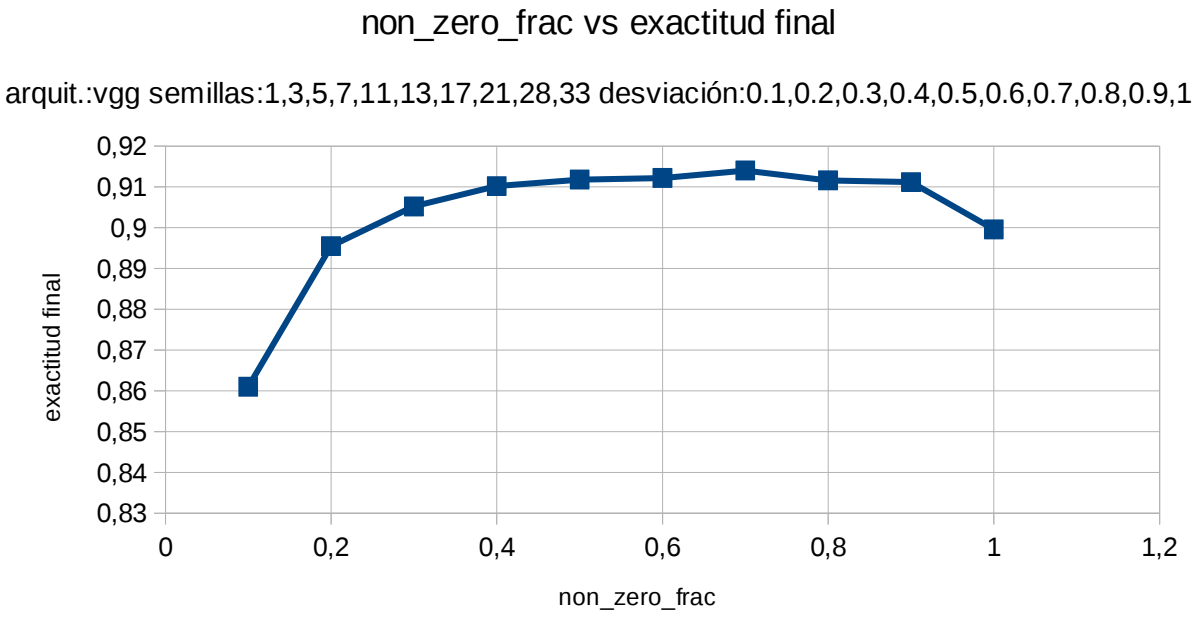


Figura 3. non_zero_frac vs exactitud final en VGG para todos las desviaciones estándar.

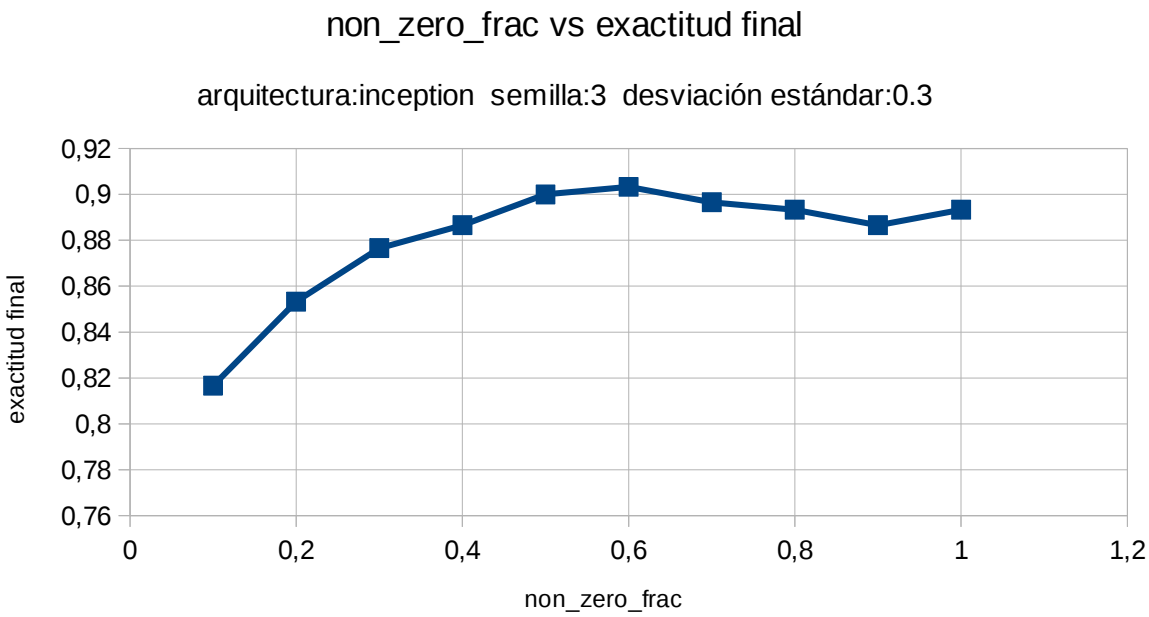


Figura 4. non_zero_frac vs exactitud final en Inception para desv.:0.3

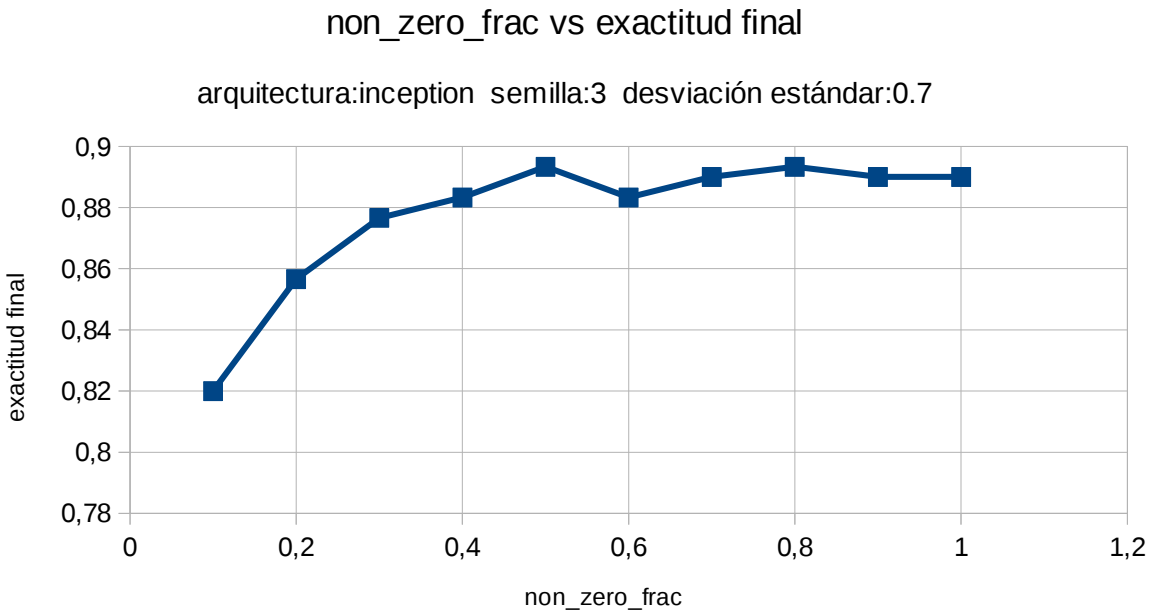


Figura 5. non_zero_frac vs exactitud final en Inception para desv.:0.7

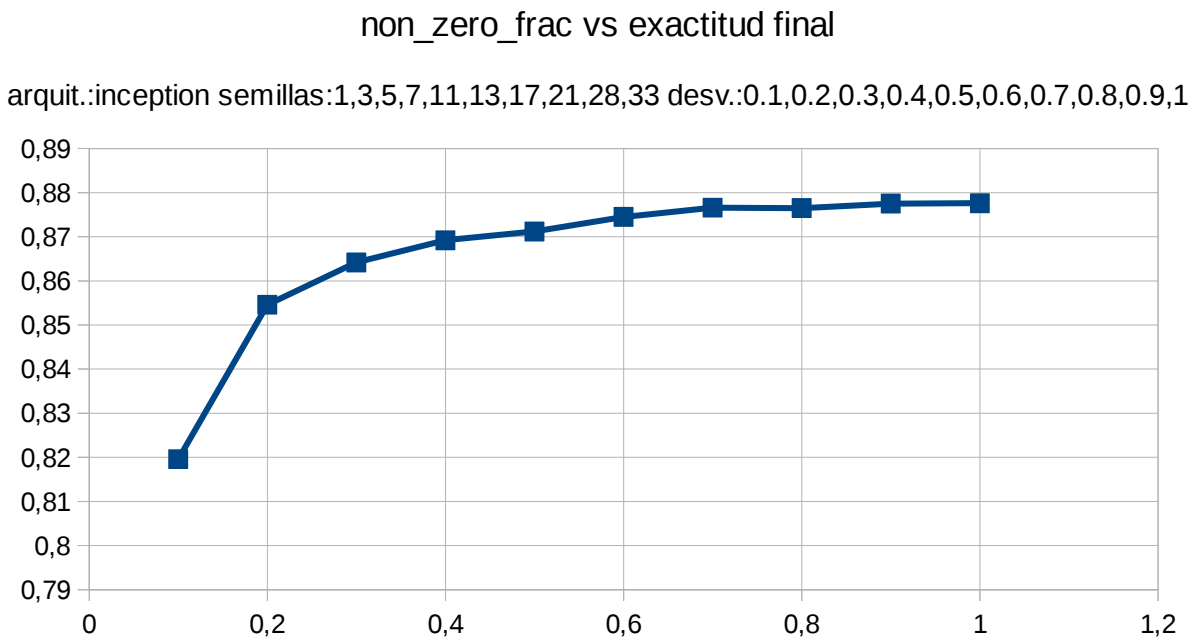


Figura 6. non_zero_frac vs exactitud final en Inception para todas las desviaciones estándar.

Con el hiperparámetro non_zero_frac se introduce no linealidad a la red y por esa razón el desempeño de la red mejora en la mayoría de los experimentos. Al promediar el efecto de non_zero_frac para todas las desviaciones estándar, en la red Inception, se observa que la no linealidad no mejora el desempeño. No obstante, se deduce de los otros experimentos en Inception que asignando un valor adecuado para la desviación estándar posibilita que la no linealidad mejore el desempeño.

Las siguientes tablas y gráficas presentan los resultados de cambiar el hiperparámetro desviación estándar en las capas densas, de las redes preentrenadas. La modificación de la desviación estándar se realizó para non_zero_frac:0.7, non_zero_frac:0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 en InceptionV3 y VGG19.

Tabla 9. Modificación de la desviación estándar en VGG19 para non_zero_frac:0.7

1 capa oculta de 1000 neuronas semilla de entrenamiento:3 semilla de orden de la BD:3 train_batch: 128 arquitectura:vgg19 non_zero_frac: 0.7 init batch:256						
desv.	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,72	0,9166	21	0,171	0,92	0,0217
0,2	0,72	0,9033	4,8899	0,1371	0,9066	0,0255
0,3	0,72	0,92	1,7388	0,0849	0,9133	0,0162
0,4	0,72	0,91	1,5923	0,1025	0,9266	0,0223
0,5	0,72	0,9	1,0151	0,1216	0,9033	0,0297
0,6	0,72	0,8933	0,5966	0,1472	0,8966	0,0405
0,7	0,72	0,9	0,5494	0,171	0,8966	0,0522
0,8	0,72	0,8866	0,5581	0,1938	0,8933	0,0636
0,9	0,72	0,8833	0,5706	0,2151	0,8966	0,0747
1	0,72	0,8766	0,5835	0,2344	0,9	0,0855

Tabla 10. Modificación de la desviación estándar en VGG19 para non_zero_frac:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9

1 capa oculta de 1000 neuronas semillas de entrenamiento:1,3,5,7,11,13,17,21,28,33 semilla de orden de la BD:3 train_batch: 128 arquitectura:vgg19 non_zero_frac:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9 init batch:256						
desv.	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,674	0,9116	20,61	0,1274	0,9138	0,0179
0,2	0,674	0,9078	4,1744	0,0946	0,9133	0,017
0,3	0,674	0,9018	2,3048	0,0922	0,9123	0,017
0,4	0,674	0,8936	1,6596	0,1138	0,9096	0,0221
0,5	0,674	0,8881	1,3119	0,1484	0,9057	0,0318
0,6	0,674	0,8822	1,0964	0,1873	0,9026	0,0447
0,7	0,674	0,8774	0,9981	0,2257	0,8994	0,0596
0,8	0,674	0,8725	0,9359	0,2617	0,8958	0,0755
0,9	0,674	0,8688	0,9183	0,2943	0,893	0,0918
1	0,674	0,8634	0,9194	0,3242	0,8907	0,1079

Tabla 11. Modificación de la desviación estándar en Inceptionv3 para non_zero_frac:0.7

1 capa oculta de 1000 neuronas semilla de entrenamiento:3 semilla de orden de la BD:3 train_batch: 128 arquitectura:inceptionV3 non_zero_frac: 0.7 init batch:256						
desv.	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,83	0,8933	0,4933	0,1842	0,9066	0,0614
0,2	0,83	0,8833	0,5229	0,3089	0,9	0,1517
0,3	0,83	0,87	0,5413	0,3794	0,8966	0,2181
0,4	0,83	0,8633	0,5485	0,4191	0,8933	0,2604
0,5	0,83	0,86	0,5513	0,4386	0,8866	0,2836
0,6	0,83	0,86	0,5522	0,4448	0,89	0,2921
0,7	0,83	0,86	0,5519	0,4426	0,89	0,291
0,8	0,83	0,86	0,5509	0,4355	0,89	0,2839
0,9	0,83	0,8633	0,5494	0,4257	0,8933	0,2737
1	0,83	0,8666	0,5475	0,4145	0,8966	0,262

Tabla 12. Modificación de la desviación estándar en Inceptionv3 para non_zero_frac:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9

1 capa oculta de 1000 neuronas semillas de entrenamiento:1,3,5,7,11,13,17,21,28,33 semilla de orden de la BD:3 train_batch: 128 arquitectura:inceptionV3 non_zero_frac:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9 init batch:256						
desv.	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,7993	0,8704	0,5493	0,2434	0,8904	0,0914
0,2	0,7993	0,8482	0,578	0,382	0,8765	0,2087
0,3	0,7993	0,8351	0,5919	0,4496	0,8648	0,2818
0,4	0,7993	0,8284	0,5969	0,4823	0,8575	0,3225
0,5	0,7993	0,825	0,5986	0,4952	0,8552	0,3403
0,6	0,7993	0,8252	0,5987	0,4964	0,8558	0,3427
0,7	0,7993	0,8264	0,5979	0,4908	0,8574	0,3358
0,8	0,7993	0,8294	0,5965	0,4813	0,8603	0,324
0,9	0,7993	0,8319	0,5948	0,4696	0,8639	0,3098
1	0,7993	0,8345	0,5927	0,457	0,867	0,2949

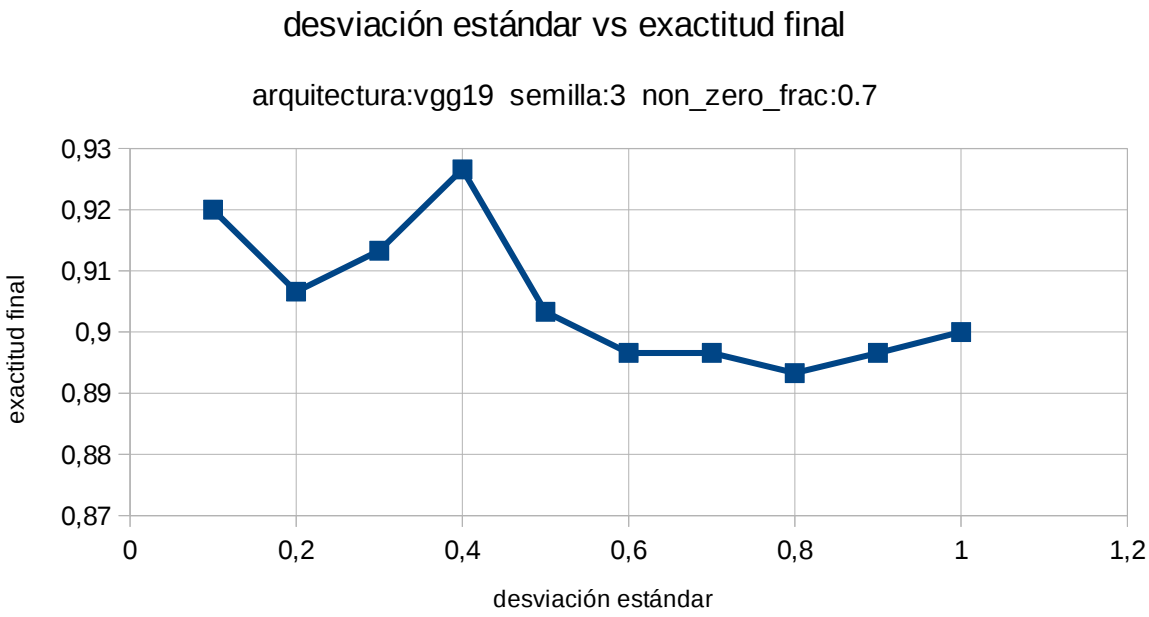


Figura 7. desviación estándar vs exactitud final en VGG19 para non_zero_frac:0.7

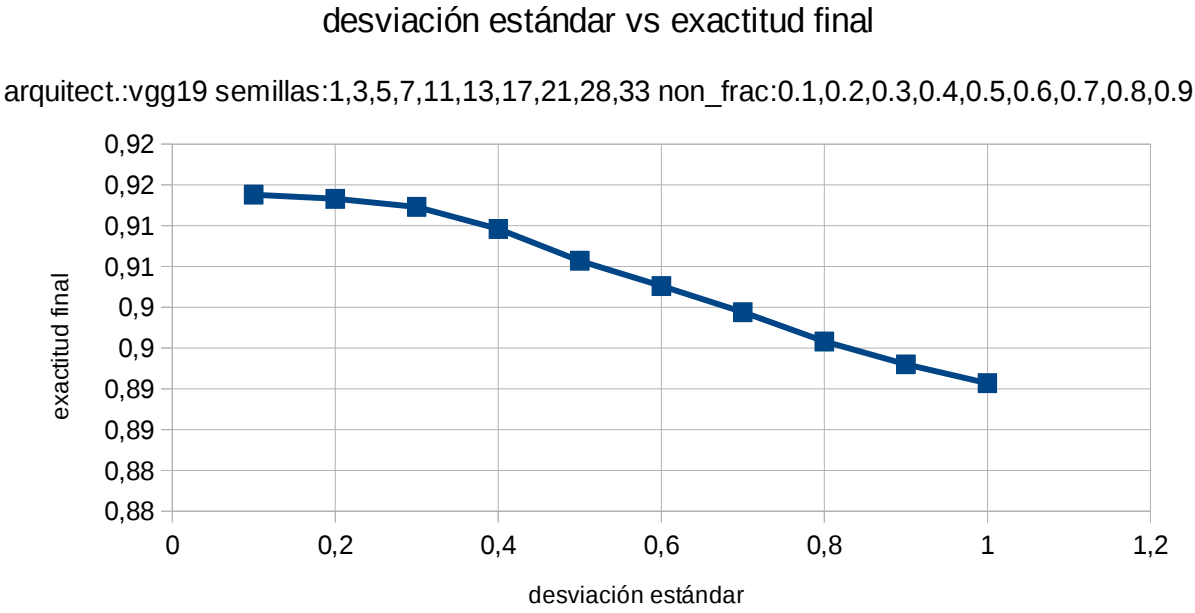


Figura 8. desviación estándar vs exactitud final en VGG19 para non_zero_frac:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9

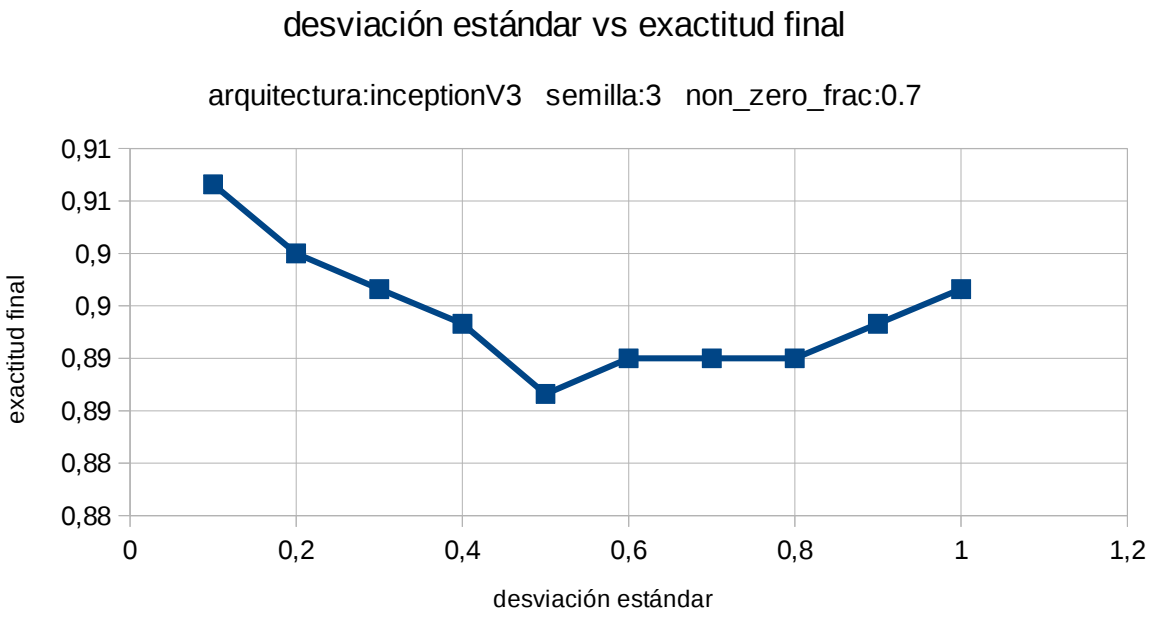


Figura 9. desviación estándar vs exactitud final en Inception para non_zero_frac:0.7

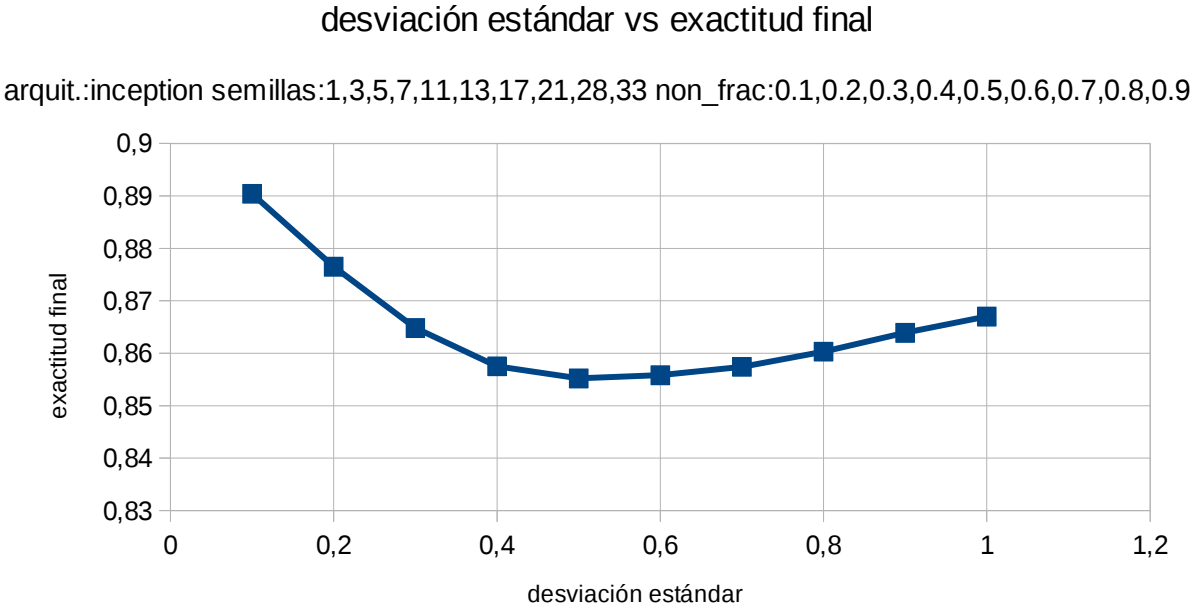


Figura 10. desviación estándar vs exactitud final en Inception para non_zero_frac:0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9

Un valor adecuado para la desviación estándar se encuentra en el intervalo 0.1 a 0.3, para Inception, y de 0.1 a 0.4 para VGG. Es evidente que se obtienen mejores resultados en los intervalos propuestos que utilizando una desviación estándar igual a 1.

Las siguientes tablas y gráficas exhiben los resultados de modificar el número de neuronas en las capas densas, de las redes preentrenadas. Los experimentos se realizaron fijando non_zero_frac a 0.5 y desviación estándar a 0.3

tabla 13. Modificación de unidades ocultas en VGG19

arquitectura:vgg19 semilla de entrenamiento:3 semilla de orden:3 non_zero_frac:0.5 train_batch:128 desviación estándar:0.3 init_batch:256						
neuronas	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
32	0,6566	0,9066	0,9765	0,2319	0,9033	0,0827
64	0,6866	0,91	0,8443	0,1692	0,9233	0,0528
128	0,6733	0,9	2,19	0,1644	0,9066	0,0341
256	0,3633	0,85	45,99	0,2879	0,8933	0,022
512	0,7166	0,91	0,9586	0,0774	0,9233	0,0146
1024	0,7433	0,9133	1,03	0,0793	0,9166	0,0153
2048	0,7133	0,9033	1,01	0,08	0,9166	0,0157

tabla 14. Modificación de unidades ocultas en Inceptionv3

arquitectura:inception semilla de entrenamiento:3 semilla de orden:3 non_zero_frac:0.5 train_batch:128 desviación estándar:0.3 init_batch:256						
neuronas	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
32	0,62	0,8133	0,9481	0,54	0,8666	0,2926
64	0,7533	0,8266	0,7831	0,4939	0,87	0,2716
128	0,64	0,83	0,9749	0,4457	0,84	0,1852
256	0,4033	0,83	2,45	0,3482	0,86	0,026
512	0,73	0,8633	0,7042	0,4396	0,8833	0,1992
1024	0,8366	0,8566	0,5221	0,4103	0,8866	0,235
2048	0,8433	0,8733	0,5208	0,4384	0,8933	0,2895

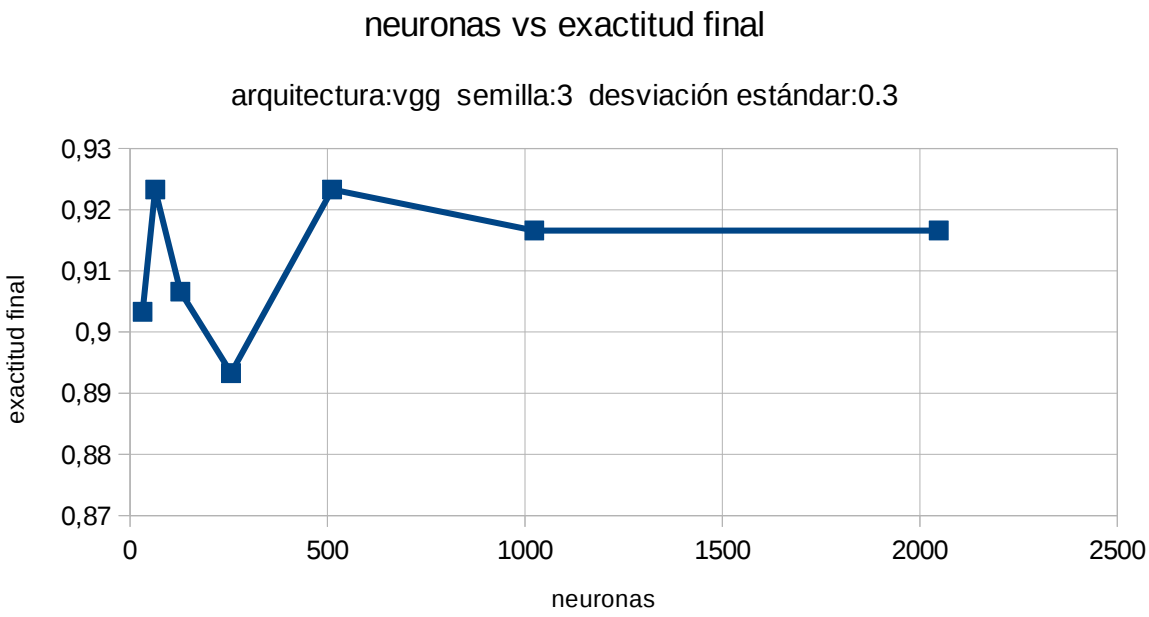


Figura 11. neuronas vs exactitud final en VGG

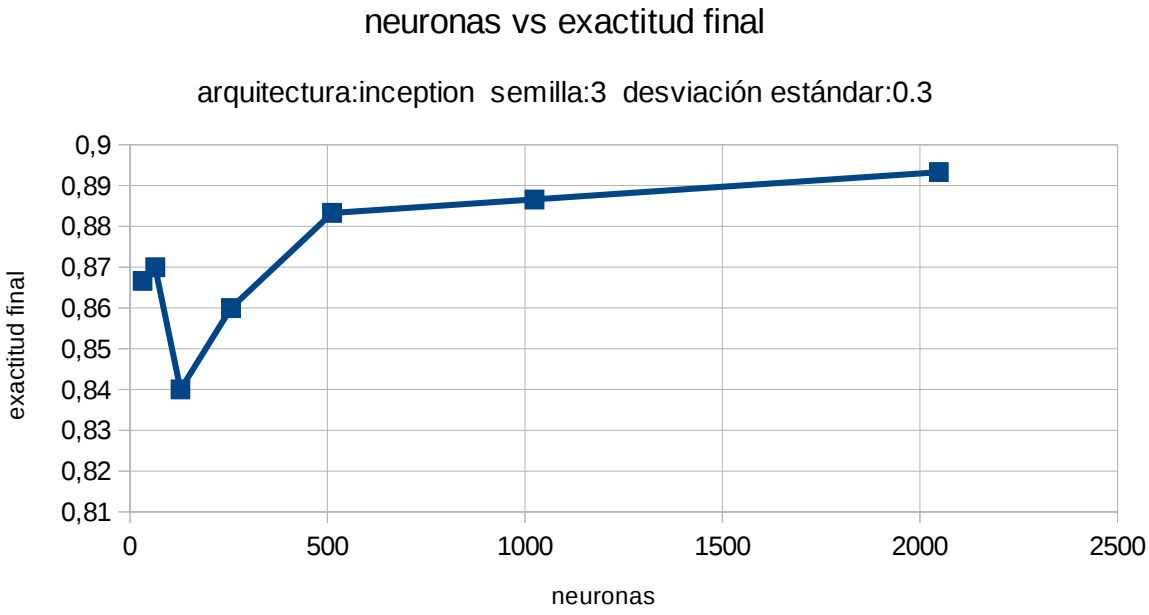


Figura 12. neuronas vs exactitud final en Inception

Un numero adecuado de neuronas, para VGG e Inception, es 512 ya que presenta buena exactitud y no implica demasiado costo computacional.

Las siguientes tablas y gráficas evidencian el efecto del nuevo esquema y los esquemas clásicos en las capas densas, de las redes preentrenadas. Los experimentos se realizaron para neuronas:32, neuronas:512

tabla 15. Comparación de esquemas en VGG para 32 neuronas ocultas

32 neuronas ocultas arquitectura:vgg semillas de entrenamiento:1,3,5,7,11,13,17,21,28,33 semilla de orden:3 non_zero_frac:0.5 train_batch:128 desviación estándar:0.3 init_batch:256			
scheme	Exactitud en época 1	Exactitud en época 25	Exactitud en época 50
our	0,8319	0,9056	0,909
lsuv	0,56	0,846	0,866
He	0,4783	0,8296	0,8526
lsuv xavier	0,7713	0,9073	0,909
xavier	0,4493	0,8163	0,8456

tabla 16. Comparación de esquemas en Inception para 32 neuronas ocultas

32 neuronas ocultas arquitectura:inception semillas de entrenamiento:1,3,5,7,11,13,17,21,28,33 semilla de orden:3 non_zero_frac:0.5 train_batch:128 desviación estándar:0.3 init_batch:256			
scheme	Exactitud en época 1	Exactitud en época 25	Exactitud en época 50
our	0,6916	0,845	0,8693
lsuv	0,2906	0,8129	0,8526
He	0,287	0,7733	0,8336
lsuv xavier	0,2333	0,7526	0,8183
xavier	0,2419	0,7813	0,841

tabla 17. Comparación de esquemas en VGG para 512 neuronas ocultas

512 neuronas ocultas arquitectura:vgg semillas de entrenamiento:1,3,5,7,11,13,17,21,28,33 semilla de orden:3 non_zero_frac:0.5 train_batch:128 desviación estándar:0.1 init_batch:256			
scheme	Exactitud en época 1	Exactitud en época 25	Exactitud en época 50
our	0,8229	0,9143	0,917
lsuv	0,6043	0,8626	0,8806
He	0,5976	0,867	0,8813
lsuv xavier	0,7953	0,9083	0,911
xavier	0,6256	0,8696	0,8779

tabla 18. Comparación de esquemas en Inception para 512 neuronas ocultas

512 neuronas ocultas arquitectura:inception semillas de entrenamiento:1,3,5,7,11,13,17,21,28,33 semilla de orden:3 non_zero_frac:0.5 train_batch:128 desviación estándar:0.1 init_batch:256			
scheme	Exactitud en época 1	Exactitud en época 25	Exactitud en época 50
our	0,8356	0,8933	0,8933
lsuv	0,3733	0,8423	0,8763
He	0,27	0,8156	0,8536
lsuv xavier	0,2726	0,8019	0,847
xavier	0,2986	0,8116	0,8553

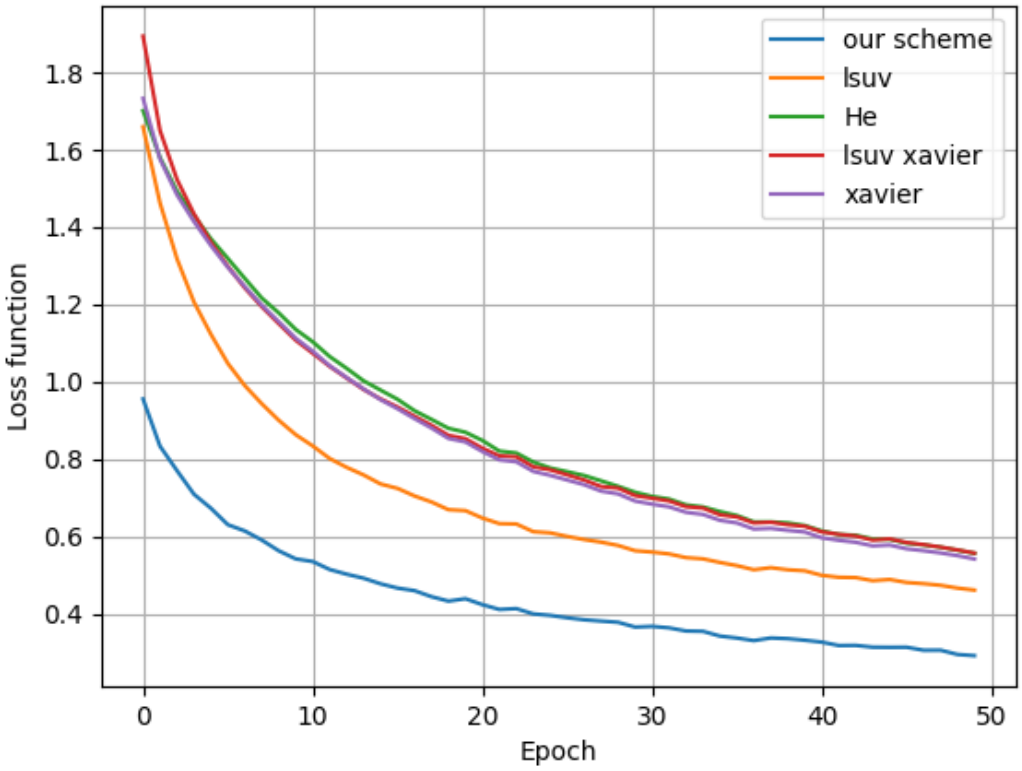


Figura 13. Función de costo para Inception con 32 neuronas ocultas

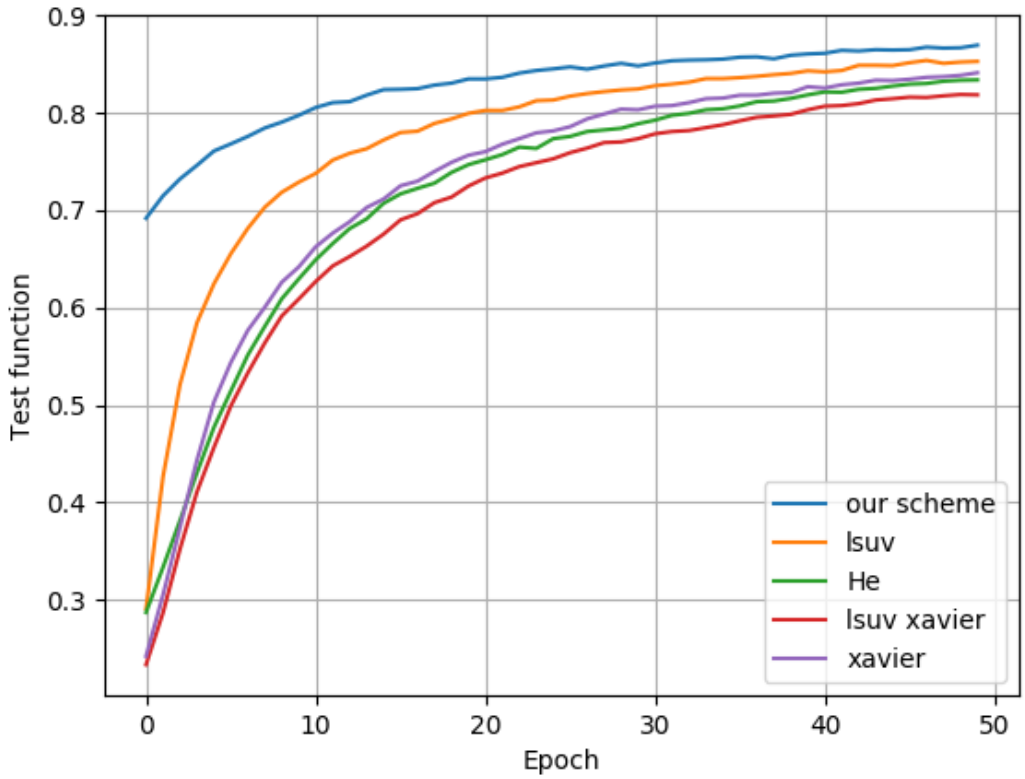


Figura 14. Función de exactitud para Inception con 32 neuronas ocultas

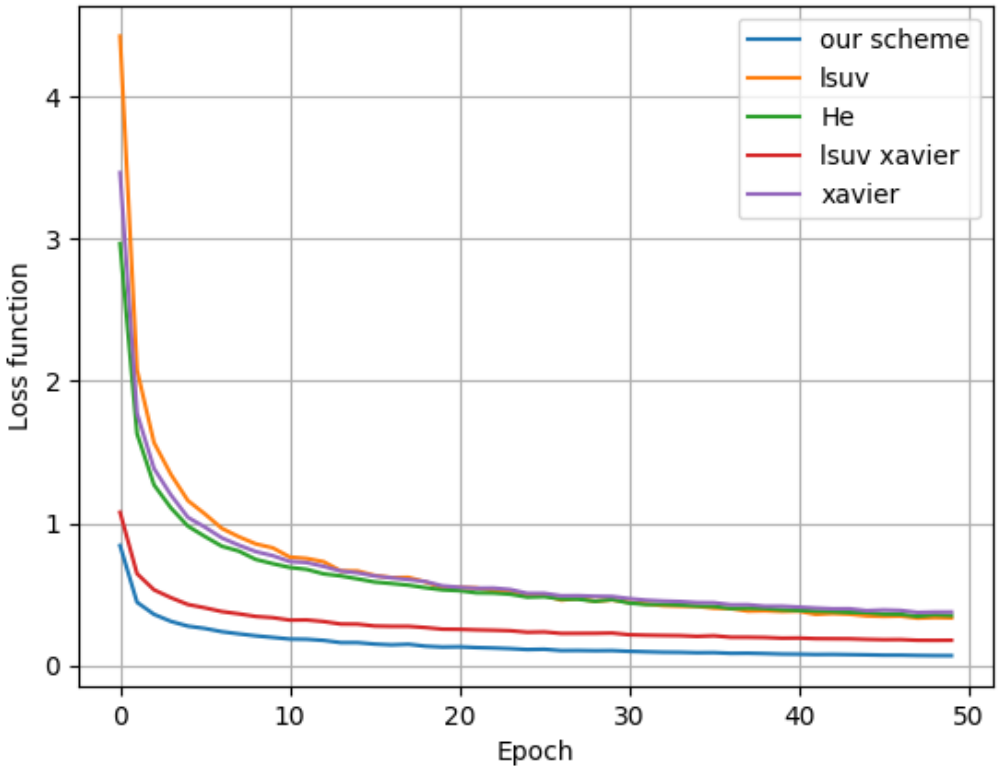


Figura 15. Función de costo para VGG con 32 neuronas ocultas

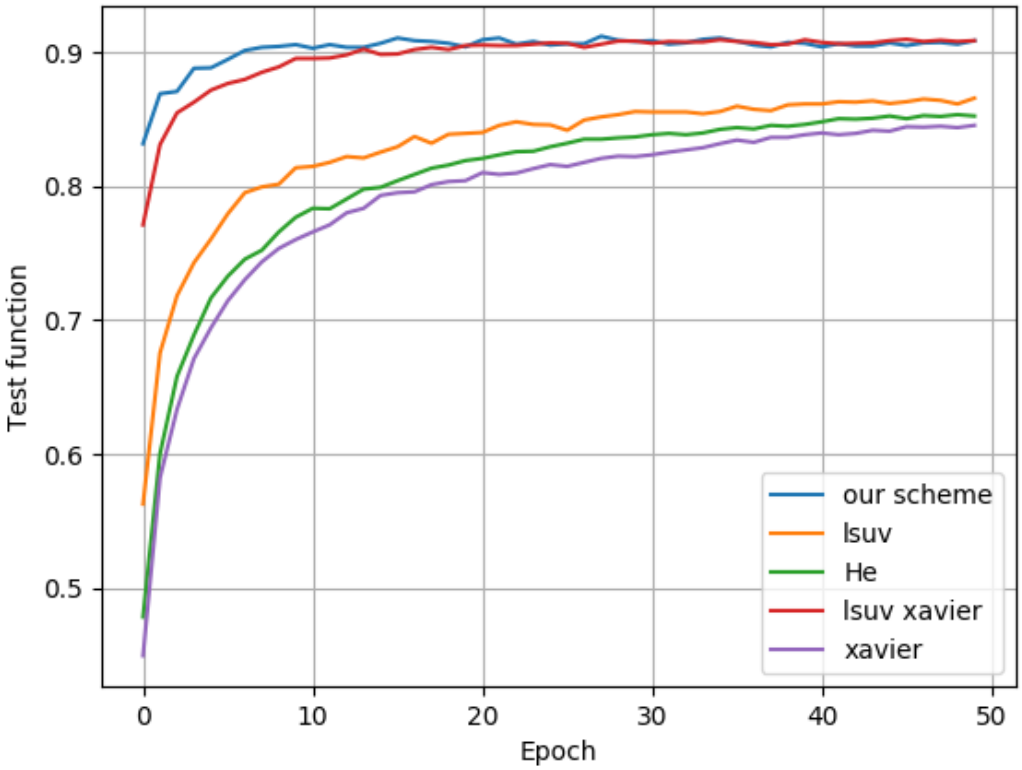


Figura 16. Función de exactitud para VGG con 32 neuronas ocultas

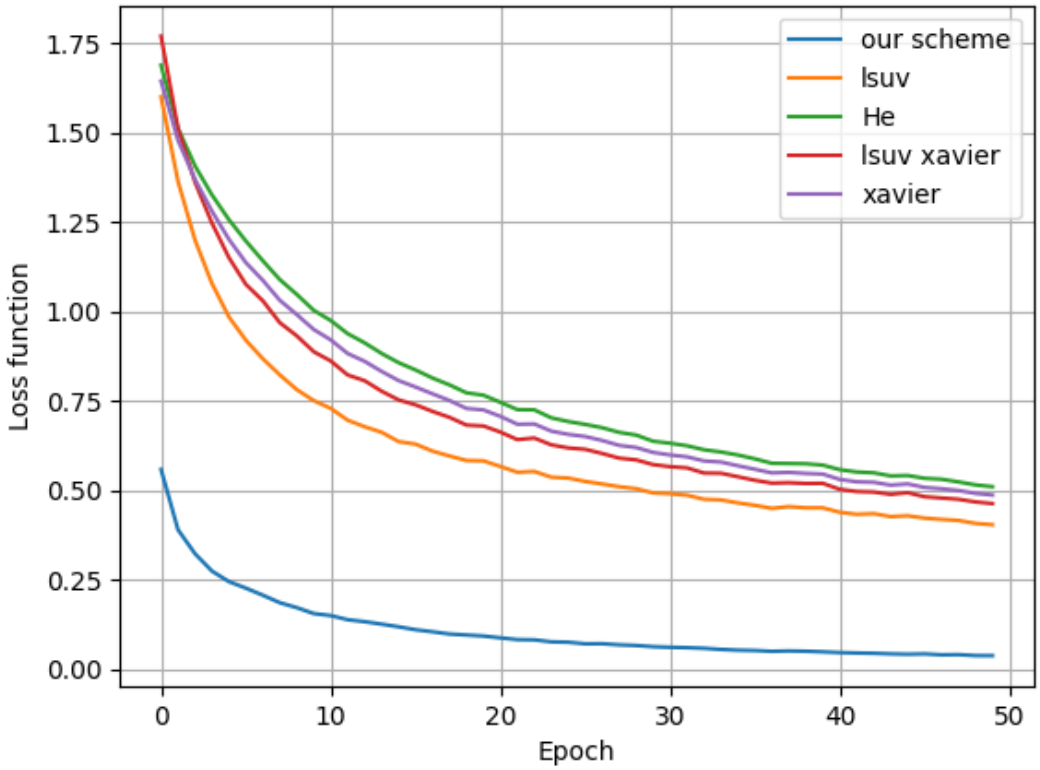


Figura 17. Función de costo para Inception con 512 neuronas ocultas

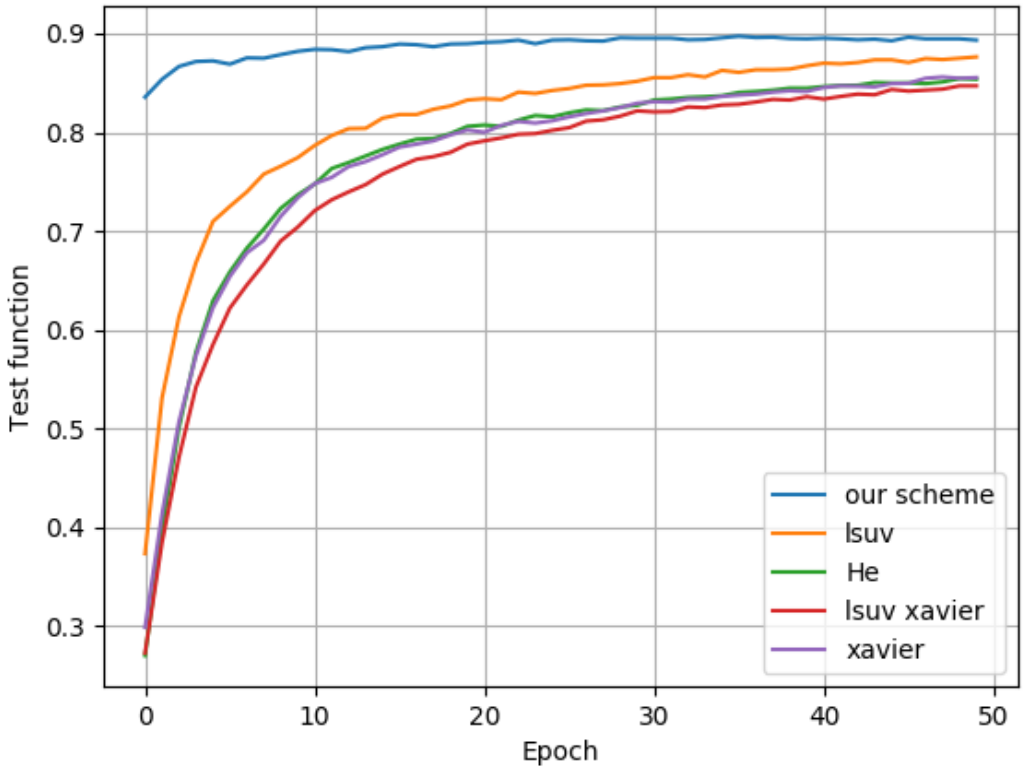


Figura 18. Función de exactitud para Inception con 512 neuronas ocultas

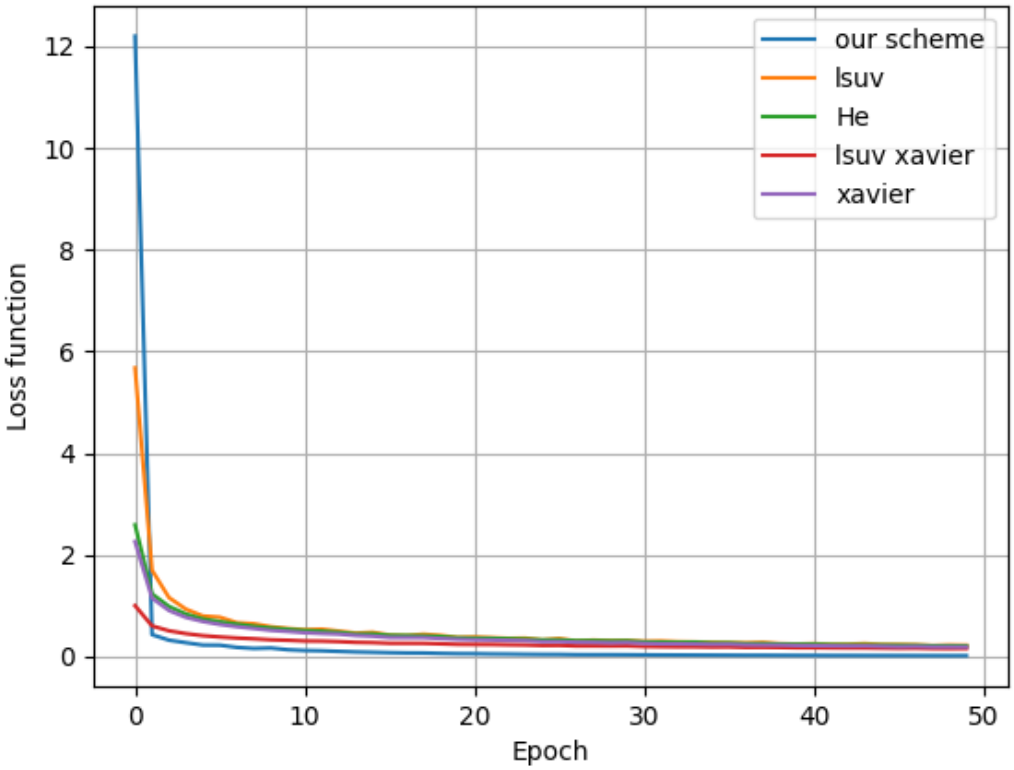


Figura 19. Función de costo para VGG con 512 neuronas ocultas

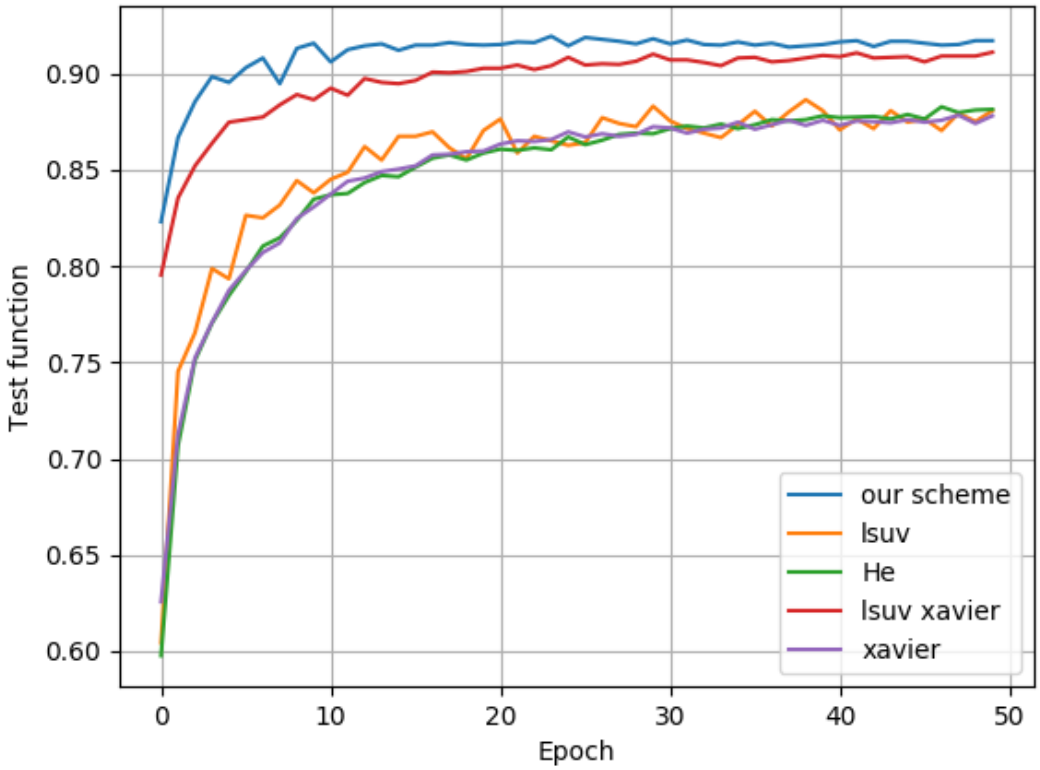


Figura 20. Función de exactitud para VGG con 512 neuronas ocultas

El nuevo esquema hace que la función de costo converja más rápidamente que en los otros esquemas y por consiguiente reduce el tiempo de entrenamiento. El esquema también posibilita que la red generalice mejor, como se refleja en las gráficas anteriores.

Las siguientes tablas y gráficas exhiben los resultados de modificar el hiperparámetro non_zero_frac en las capas convolucionales de la red neuronal. Los experimentos se realizaron inicializando los filtros convolucionales con las técnicas Gram-Schmidt y filtros de Gabor. Los filtros de Gabor utilizados tienen las siguientes especificaciones: desviación estándar de la envoltura gaussiana=4, longitud de onda senoidal=10, tamaño del filtro=3*3, ángulos entre 0 y 180 grados. En los experimentos se fija non_zero_frac:0.5, desviación estándar:0.3 en las capas densas y desviación estándar:0.5 para las capas convolucionales.

tabla 19. Modificación de non_zero_frac con la técnica Gram-Schmidt

semilla de entrenamiento:3 semilla de orden de la BD:3 técnica:Gram-Schmidt train_batch: 128 non_zero_frac en capas densas:0.5 desviación estándar en capas densas:0.3 desviación estándar en capas convolucionales:0.5 init batch:256						
non_zero_frac	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,32	0,55	1,34	0,37	0,57	0,09
0,2	0,32	0,56	1,32	0,26	0,61	0,02
0,3	0,32	0,59	1,36	0,3	0,6	0,02
0,4	0,32	0,54	1,47	0,62	0,63	0,05
0,5	0,33	0,59	1,77	0,86	0,63	0,14
0,6	0,33	0,57	1,59	0,89	0,61	0,13
0,7	0,35	0,45	1,95	1,18	0,57	0,46
0,8	0,33	0,49	3,31	1,1	0,59	0,57
0,9	0,35	0,31	3,54	1,53	0,38	1,36
1	0,35	0,25	24,85	1,57	0,32	1,47

tabla 20. Modificación de non_zero_frac con la técnica filtros Gabor

semilla de entrenamiento:3 semilla de orden de la BD:3 técnica:filtros Gabor train_batch: 128 non_zero_frac en capas densas:0.5 desviación estándar en capas densas:0.3 desviación estándar en capas convolucionales:0.5 init batch:256						
non_zero_frac	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,26	0,3	1,64	1,34	0,307	1
0,2	0,25	0,32	1,58	1,32	0,31	0,91
0,3	0,28	0,32	1,55	1,3	0,35	0,94
0,4	0,25	0,33	1,53	1,31	0,34	0,94
0,5	0,26	0,33	1,59	1,31	0,35	0,92
0,6	0,28	0,34	1,99	1,37	0,38	1,04
0,7	0,25	0,34	2,08	1,43	0,36	1,21
0,8	0,26	0,33	2,55	1,46	0,33	1,29
0,9	0,27	0,34	4,87	1,45	0,37	1,31
1	0,27	0,23	8,51	1,6	0,23	1,6

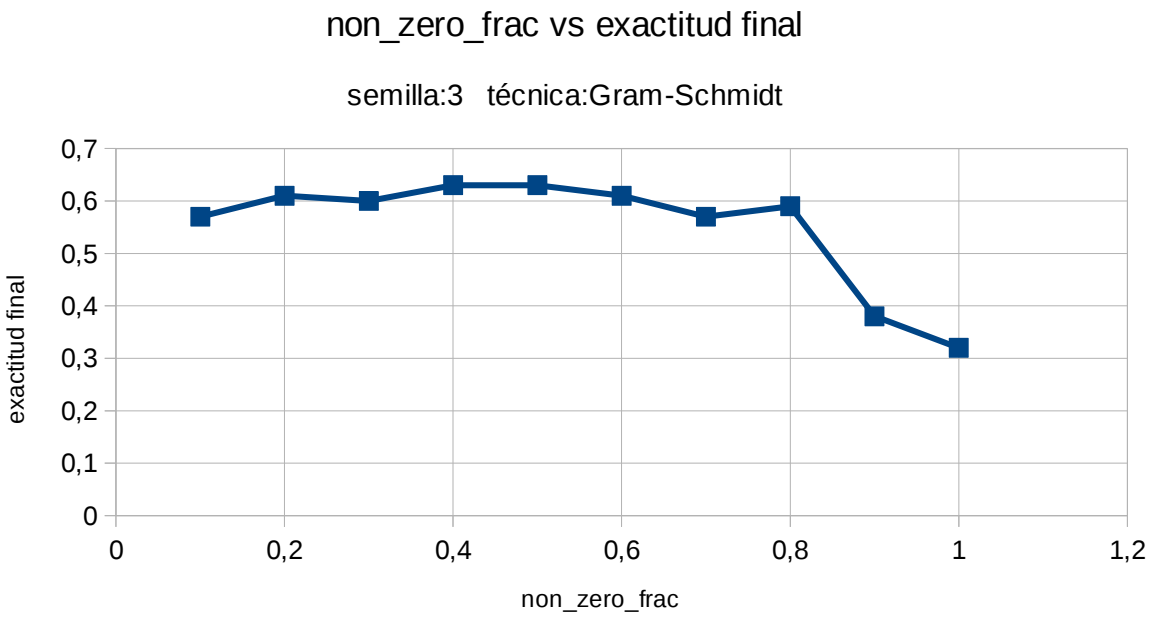


Figura 21. non_zero_frac vs exactitud final con la técnica Gram-Schmidt

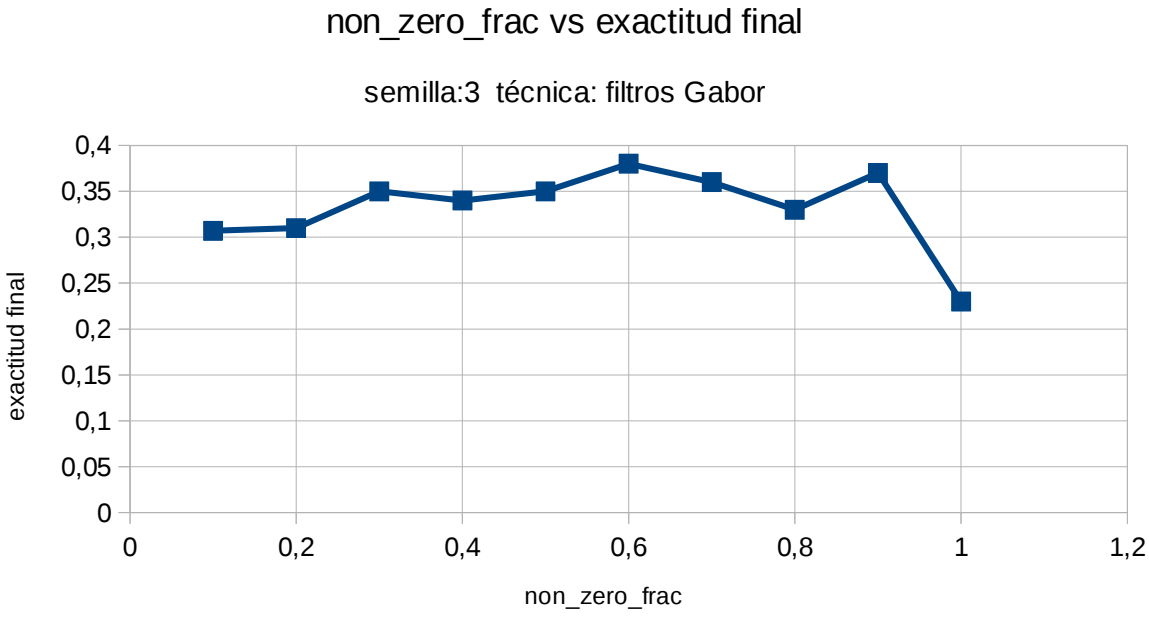


Figura 22. non_zero_frac vs exactitud final con la técnica filtros Gabor

Es evidente que un valor adecuado para non_zero_frac, en capas convolucionales, se encuentra en el intervalo 0.5 a 0.8 para ambas técnicas. También es notorio que en esta red el desempeño obtenido con la técnica Gram-Schmidt supera el obtenido con filtros de Gabor. Introduciendo no linealidad en las capas convolucionales se mejora el desempeño de esta red, en ambas técnicas.

Las siguientes tablas y gráficas muestran el efecto de modificar la desviación estándar en las capas convolucionales de la red. En los experimentos se fija non_zero_frac:0.5, desviación estándar:0.3 en las capas densas y non_zero_frac:0.4 para las capas convolucionales.

tabla 21. Modificación de la desviación estándar con la técnica Gram-Schmidt

semilla de entrenamiento:3 semilla de orden de la BD:3 técnica:Gram-Schmidt train_batch: 128 non_zero_frac en capas densas:0.5 non_zero_frac en capas convolucionales:0.4 desviación estándar en capas densas:0.3 init batch:256						
desviación estándar	Exactitud antes de entrenar	Exactitud en época 10	costo en época 1	costo en época 10	exactitud final	costo final
0,1	0,32	0,49	1,47	0,97	0,57	0,42
0,2	0,32	0,55	1,39	0,7	0,61	0,15
0,3	0,32	0,57	1,36	0,55	0,6	0,07
0,4	0,32	0,58	1,39	0,52	0,64	0,04
0,5	0,32	0,54	1,47	0,6	0,63	0,05
0,6	0,32	0,6	1,56	0,75	0,64	0,06
0,7	0,32	0,58	1,88	0,83	0,66	0,08
0,8	0,32	0,57	2,84	0,81	0,65	0,1
0,9	0,32	0,58	3,46	0,87	0,66	0,15
1	0,32	0,52	5,13	1,03	0,66	0,14

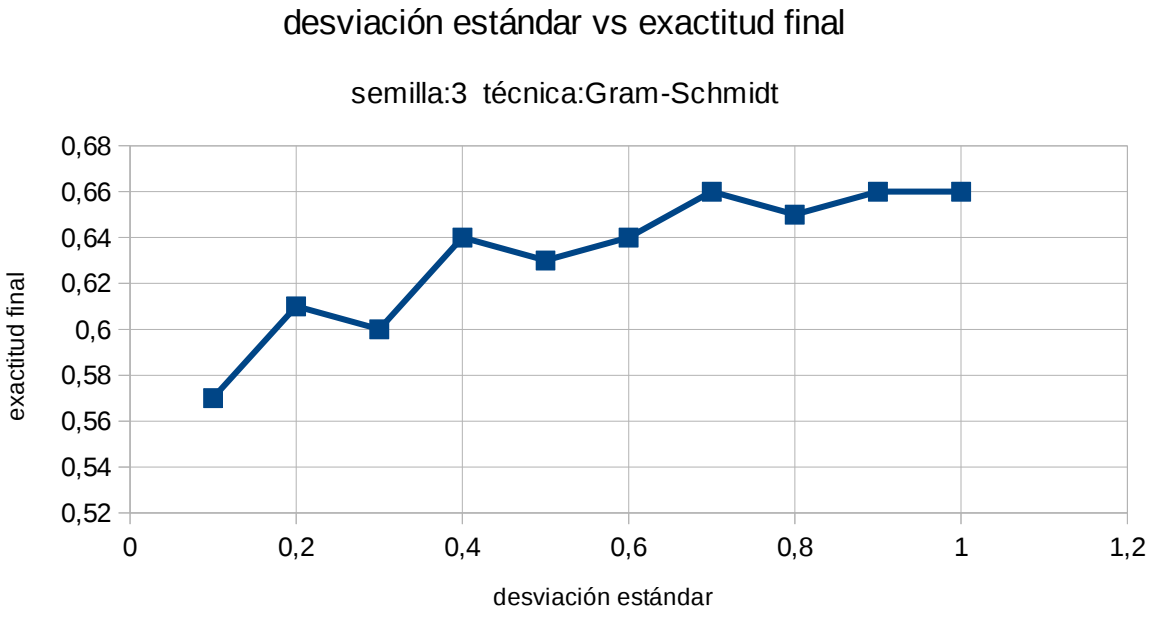


Figura 23. desviación estándar vs exactitud final con la técnica Gram-Schmidt

Un valor adecuado para la desviación estándar en capas convolucionales se encuentra en el intervalo 0.7 a 1. No se cumple el mismo intervalo que en las

capas densas de las redes Inception y VGG. No obstante, se puede concluir, por los experimentos, que es mejor utilizar una desviación estándar menor a uno.
Las siguientes tablas y gráficas evidencian el efecto del nuevo esquema y los esquemas clásicos en la red convolucional.

tabla 22. Comparación de esquemas en la red neuronal

semillas de entrenamiento:1,3,5,7,11,13,17,21,28,33 semilla de orden de la BD:3 técnica:Gram-Schmidt train_batch: 128 non_zero_frac en capas densas:0.5 non_zero_frac en capas convolucionales:0.4 train_batch:128 desviación estándar en capas densas:0.3 desviación en capas convolucionales:0.7 init batch:256			
Esquema	Exactitud en época 1	Exactitud en época 25	Exactitud en época 50
our	0.3894	0.6172	0.6461
lsuv	0.2516	0.3689	0.4475
He	0.3309	0.5451	0.5808
lsuv xavier	0.2904	0.5232	0.5436
xavier	0.2138	0.4404	0.5167

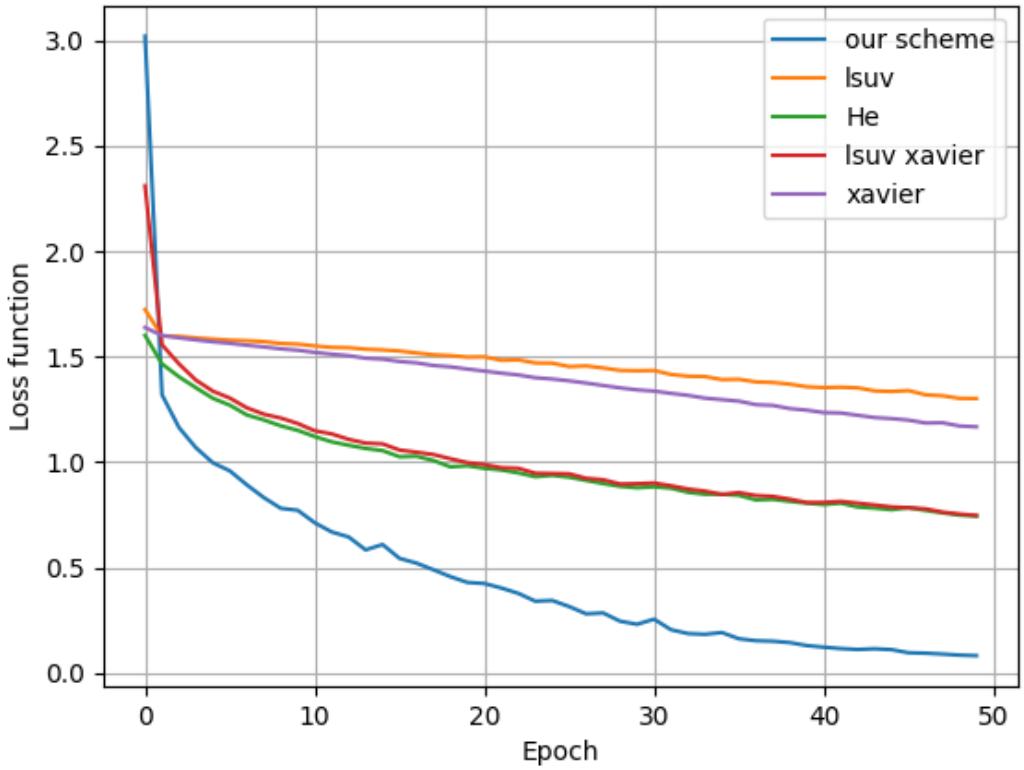


Figura 24. Función de costo de la red neuronal con diferentes esquemas

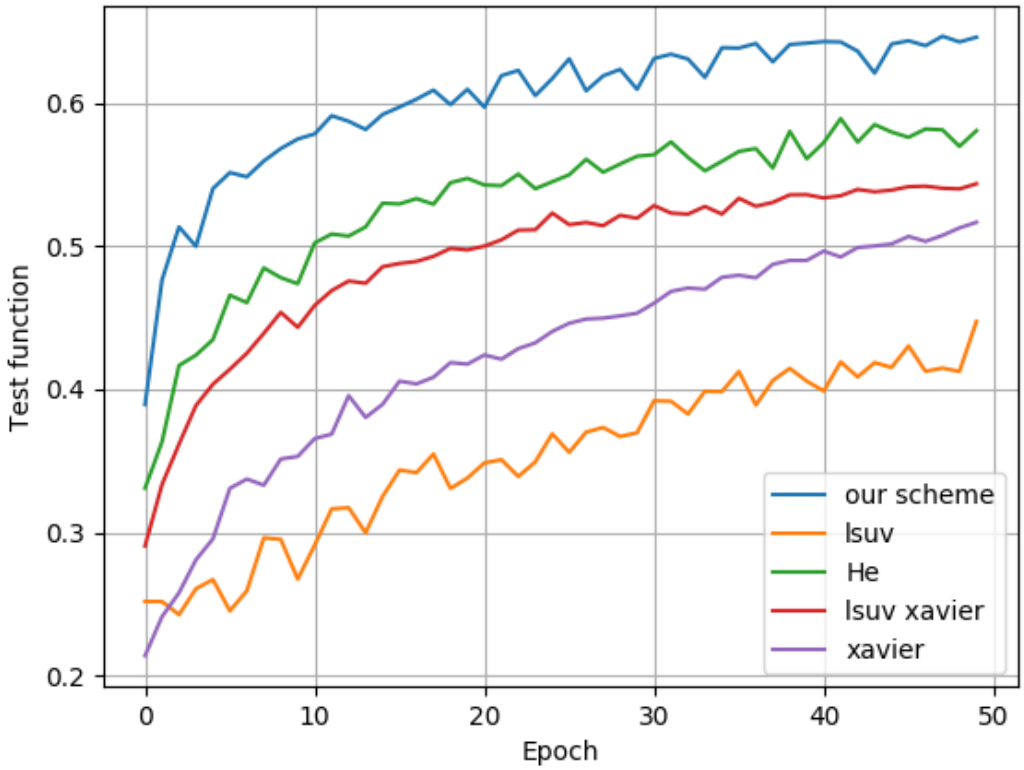


Figura 25. Función de exactitud de la red neuronal con diferentes esquemas

El nuevo esquema hace que la función de costo converja más rápidamente que en los otros esquemas y por consiguiente reduce el tiempo de entrenamiento. El esquema también posibilita que la red generalice mejor al final del entrenamiento.

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

Conclusiones:

El nuevo esquema hace que la función de costo converja más rápido que en los esquemas clásicos.

La exactitud final obtenida con el nuevo esquema supera la exactitud final de los esquemas clásicos.

El nuevo esquema requiere menos tiempo de entrenamiento que los esquemas clásicos.

Aumentando el valor de init_batch aumenta la exactitud final y disminuye el costo final solo si las imágenes seleccionadas, para init_batch, son suficientemente representativas del fenómeno físico analizado.

En todos los experimentos se encontró que un valor adecuado para non_zero_frac, en capas densas y convolucionales, está en el intervalo 0.5 a 0.8.

Introducir no linealidad a la red mejora su desempeño solo si se elige un valor adecuado para la desviación estándar.

La exactitud más alta, en todos los experimentos, se obtuvo con una desviación estándar menor a uno.

El intervalo adecuado para la desviación estándar en las capas densas es diferente al intervalo en las capas convolucionales.

Se obtienen mejores resultados utilizando la técnica Gram-Schmidt, en las capas convolucionales, que usando filtros de Gabor con las especificaciones empleadas.

Recomendaciones:
El valor de init_batch no debe ser igual al número de neuronas en la última capa oculta, en caso de que sean iguales el algoritmo Moore Penrose encuentra una solución que se sobre ajusta a las imágenes de init_batch.

Comparar la técnica Gram-Schmidt con los filtros de Gabor utilizando múltiples especificaciones.

Trabajo futuro:
Determinar el valor optimo del init_batch y la desviación estándar teniendo en cuenta la arquitectura de la red y el fenómeno físico analizado.

Determinar matemáticamente que tan representativa es una imagen del fenómeno analizado

REFERENCIAS

Arfken, G (1985). Gram-Schmidt Orthogonalization in Mathematical Methods for Physicists. Orlando, FL: Academic Press, pages 516-520.

Chollet, F. (2018). Deep learning with python. Greenwich: Manning publications Co

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna (2015). Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567.

D. Mishkin & J. Matas (2015). All you need is a good init. Tetrahedron, 69(14):3013–3018

K. He, X. Zhang, S. Ren, & J. Sun (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pages 1026–1034.

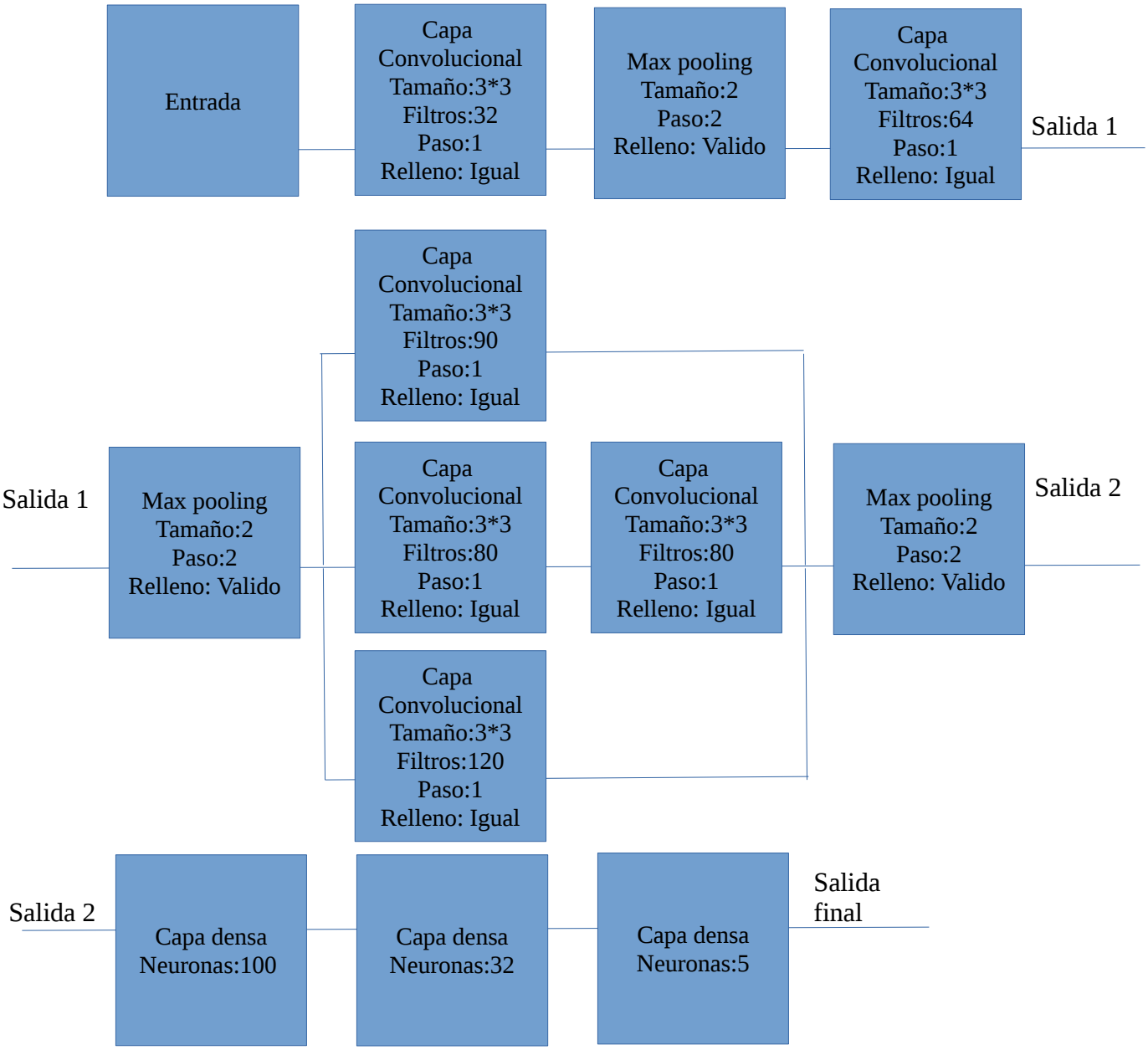
K. Simonyan & A. Zisserman (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

Nielsen, A. (2015). Neural Networks and Deep Learning. Determination Press

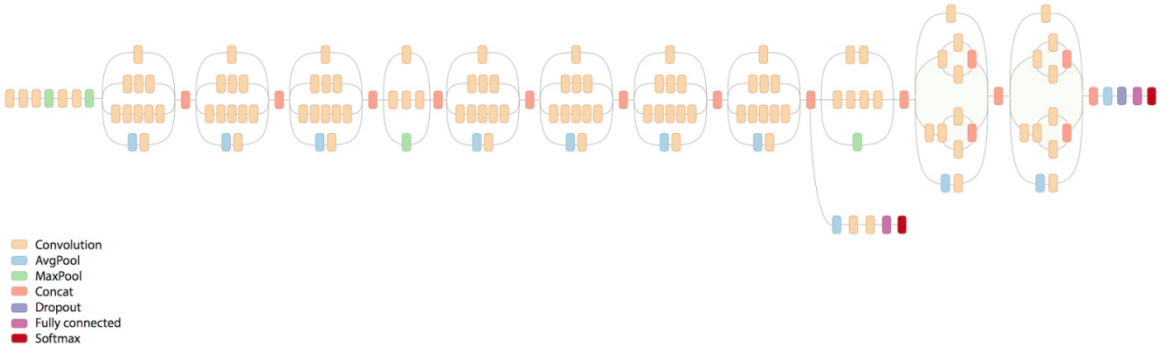
X. Glorot & Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pages 249–256.

APÉNDICE

Apéndice A



Apéndice B



Tomado de https://github.com/tensorflow/models/tree/master/research/inception

Apéndice C

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

K. Simonyan and A. Zisserman (2015). Very deep convolutional networks for large-scale image recognition.

FIRMA ESTUDIANTES Daniel Gómez Guzmán

FIRMA ASESOR

FECHA ENTREGA: Julio 26, 2018

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO____ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA ESTUDIANTES _____

_____

_____

FIRMA ASESOR _____

FECHA ENTREGA: _____

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

_____

RECHAZADO____ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____