VGGlot: Classification of Omniglot Dataset

Susana Figueroa, Javiera Fuentes & Matías Masjuan

1. Introducción

Una de las problemáticas emergentes dentro de la inteligencia artificial es medir y perfeccionar la capacidad que tienen las máquinas para generalizar ciertos patrones en base a una entrada específica. En particular, para una persona corriente que logra reconocer e identificar un cierto patrón de una imagen, le resulta sencillo poder reconocer la misma imagen aún cuando presenta ciertas modificaciones, ya sea transformaciones en el color y/o espacio. Lamentablemente, esta tarea no es necesariamente sencilla para las máquinas, donde usualmente necesitan cientos o incluso miles de ejemplares para lograr un buen desempeño ante este este mismo problema.

El *dataset* utilizado fue el de *omniglot*, el cual presenta un conjunto de imágenes de 30 alfabetos distintos, donde cada uno presenta una cantidad determinada de caracteres, siendo un total de 940 entre todos los alfabetos. Cada carácter posee un total de 20 ejemplares, los cuales servirán para entrenar, validar y evaluar cada uno de los modelos generados.

En el presente informe se buscará proponer, implementar y evaluar un nuevo mecanismo para determinar la capacidad que tienen los modelos para la problemática de generalización. Para ello, se entrenará una serie de modelos en base a una baja cantidad de ejemplares de caracteres y se evaluará su desempeño a través de las imágenes de *testing*. Finalmente, se realizarán modificaciones a los caracteres ya estudiados mediante el uso de *Data Augmentation* y se evaluará su rendimiento para estudiar la diferencia del desempeño obtenido y determinar su capacidad de generalización.

2. Revisión bibliográfica

2.1. Lecturas recomendadas

2.1.1. Shortcut Learning in Deep Neural Networks

El paper (Geirhos et al., 2020) aborda el problema de *Shortcuts* y plantea recomendaciones para detectarlos o evitarlos. Los *shortcuts* son reglas de decisión que tienen buen desempeño en los *benchmarks* habituales pero fallan al transferir el modelo a condiciones de testing más desafiantes o distintas, como escenarios en la vida real. El *shortcut learning* generalmente es detectado por una discrepancia entre la estrategia de aprendizaje prevista y la realmente aprendida, provocando un fracaso inesperado, esto se debe a que las redes neuronales no pueden inferir desde los datos de entrenamiento cuál es la solución o características que utilizar para clasificar. Un ejemplo de esto es cuando una red es entrenada para distinguir estrellas de lunas. En todas las imágenes de training la estrella aparece en la misma esquina de la imagen, lo mismo con la luna pero en una esquina opuesta, en este caso, la red aprende que la diferencia entre una clase y otra es la posición dentro de la imagen y no la forma del objeto. Ahora ¿Cómo distinguimos una *shortcut* de las otras reglas de decisión? Podemos definir una taxonomía, pensemos en todas las reglas posibles, solo algunas de ellas resuelven el set de entrenamiento, dentro de estas, solo algunas generalizan a un set the test I.I.D (independent and identically distributed), finalmente dentro de este subset de soluciones, los *shortcuts* son aquellos que fracasan en generalizar a un tipo de dato distintos (Out-of-distribution test set), mientras que las soluciones esperadas si lo hacen. Entre las sugerencias presentadas están:

1. Importancia de combinación de *features*: la definición de un objeto es una combinación de información que influencia la regla de decisión.
2. El tamaño del dataset no asegura que esté libre de *shortcuts*, el dataset es útil solo mientras sea un buen sustituto de la regla que realmente nos interesa.
3. Usar ciertas *features* crea insensibilidad a otras. Por lo que solo si las *features* son elegidas hacia una dirección correcta se podrá generalizar.
4. Para detectar *shortcuts*, es útil testear si el baseline de un modelo excede las expectativas, aún cuando no se usan las *features* intencionadas.
5. Se deben diseñar buenos tests del tipo O.O.D.
6. La arquitectura, el set de entrenamiento, la función de pérdida y de optimización son claves para determinar si se está aprendiendo un *shortcut* o no.

2.1.2. Transformers generalize differently from information stored in context vs weights

Arquitecturas basadas de *Transformers* tiene la habilidad para usar información almacenada en los pesos durante el entrenamiento (*in-weights learning*) así como también información guardada solo en los *inputs* proporcionados como contexto al momento de la inferencia (*in-context learning*). El paper (Chan et al., 2022) analiza las diferencias entre sesgos inductivos cuando un *Transformer* representa y generaliza la información proporcionada por estas dos fuentes. Para esto se adapta el paradigma de Dasgupta: *partial exposure* en donde cada estímulo tiene dos *features* en la que

sólo una de ellas predice el *label* correcto y se evalua cómo el modelo generaliza los datos *held-out*. El análisis es realizado sobre *transformers* entrenados desde cero y también sobre transformers pre-entrenados. Para transformers entrenados desde cero se obtienen las siguientes conclusiones:

- **La generalización con información de los pesos es *rule-based***: Al entrenar la red haciendo que solo almacene un mapeo de .[es]timulos-labels"sin utilizar contexto, sus predicciones son basadas en una regla de generalización.
- **La generalización con información del contexto es basada en ejemplos**: Para esto se entreno un modelo para *few-shot in-context learning*. Al evaluar el modelo sus predicciones eran basada en ejemplos.

Para modelos pre-entrenados tenemos dos resultados:

- **Modelos grandes (70B parámetros):** para dichos modelos pre-entrenados la generalización con información de contexto es parcialmente basada en reglas.
- **Modelos pequeños (1B-7B)**: Se van haciendo cada vez menos basados en reglas.

Los autores concluyen también que modelos pre-entrenados de lenguaje muestran generalizaciónes basadas en reglas debido a que el lenguaje natural está estructurado de esta forma.

2.1.3. Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks

En esta investigación (Lake y Baroni, 2018) se busca evaluar la capacidad que posee una *Recurrent Neural Network* (RNN) para comprender la composición de palabras en un texto determinado. En otras palabras, si una máquina es capaz de entender palabras como *"girar"* o *"dos veces"*, debe ser capaz de entender la composición entre ellas, tal como *"girar dos veces"*. Así mismo, debería ser capaz de entender otras reglas de generalización. Por ejemplo, si la RNN ahora conoce una nueva palabra, tal como *"saltar"*, entonces también debería ser capaz de comprender la composición *"saltar dos veces"*. Sin embargo, la conclusión ante este problema es que la RNN falla drásticamente al momento de generalizar, dada la falta de habilidades de composición sistemática.

El *dataset* utilizado fue el SCAN (***S**implified version of the **C**omm**AI** **N**avigation tasks*) el cual cuenta con una serie de instrucciones con un respectivo *target* objetivo, tal como se muestra en la siguiente tabla.

Comando	**Acción**
jump	JUMP
jump left	LTURN JUMP
turn left twice	LTURN LTURN

Tabla 1: Ejemplos de comandos de SCAN y su respectiva acción.

El modelo utilizado fue un *framework sequence-to-sequence*, en el cual dos RNN aprenden en conjunto para *mapear* las secuencias de *input* a las de *output*. En particular, una RNN trabaja como *encoder* y recibe como *input* una secuencia de palabras, representando el comando. Luego, esta representación pasa a la segunda RNN que trabaja como *decoder*, la cual genera el *output* de la secuencia acción por acción. Finalmente, el *ouput* del *decoder* se compara con el *ground truth*, y mediante un *backpropagation* va actualizando los parámetros del *encoder* y del *decoder*.

En particular, se realizaron cuatro experimentos, con los respectivos resultados:

1. **Generalizando un subconjunto de comandos aleatorios**: Al momento de generar nuevos comandos producidos de forma aleatorio, se obtienen muy buenos resultados (más del 99 % de acc). Esto considerando que varios subconjuntos de los comandos aleatorios están presentes en datos de *training*.
2. **Generalizando comandos con acciones largas**: A pesar de que el modelo entrena con varios comandos, al momento de generar secuencias de acciones largas se tienen muy malos resultados, con un 14 % de accurcy aproximadamente.
3. **Generalizando la composición de comandos primitivos**: En este experimento, se busca generalizar los comandos a partir de primitivas. Por ejemplo, si el modelo fue entrenado con comandos de *"jump"* y *"turn left"*, se esperaría que sepa generalizar comandos compuestos como *"jump left"*. Los resultados son sumamente variados dependiendo del comando generalizado, lo cual demuestra lo mal que generaliza el modelo.
4. **Composición en traducción**: Al ingresar comandos teniendo como objetivo la traducción a otro idioma, se obtienen excelentes resultados, demostrando que el modelo es capaz de generalizar para distintos idiomas al menos para pequeñas secuencias de palabras.

2.2. Lecturas adicionales

2.2.1. How transferable are features in deep neural networks?

El paper plantea que todas las redes neuronales modernas tienden a aprender en sus primeras capas *features* que se asemejan a filtros de Gabor o *color blobs*, así independiente de la función de costo y la naturaleza del dataset las primeras capas tienes *features* estándar, por lo que son llamadas *features* generales. Por otro lado, las últimas capas dependen mayoritariamente en el dataset y la tarea escogida, por lo que se denominan *features* específicas. Con este contexto, los autores (Yosinski et al., 2014) plantean las preguntas:

- ¿Se puede cuantificar el grado de generalidad o especificidad una capa de particular?
- ¿La transición ocurre bruscamente en una capa o se extiende por varias capas?
- ¿Dónde toma lugar esta transición: al inicio, en el medio o al final de la red?

El interés de estas preguntas se debe a que puede ser utilizadas en *trasnfer learning*, donde primero se entrena una red base en un dataset y task específico y luego se utilizan las *features* aprendidas en una segunda red entrenada con otro dataset y tarea, este proceso funcionará si las features son generales, es decir, apropiadas para ambos datasets y tareas. Para el estudio los autores buscan medir el grado de generalidad de un set de *features* aprendido en una tarea A midiendo como se extienden a una tarea B. Para crear ambas tareas se divide aleatoriamente el dataset ImageNet en dos grupo. Dejando 500 clases del conjunto de datos en cada grupo. Se entrenan redes convolucionales para A y B. Específicamente se entrenan dos redes, definiendo $n = 3$:

- *Selffer network B3B*: las primeras $n = 3$ capas son copiadas desde B y congeladas. Las capas 4 a 8 son inicializadas de forma aleatoria y entrenadas en el dataset B. Esta en una red de control.
- *Transfer network A3B*: las primeras $n = 3$ capas son copiadas desde A y congeladas. Las capas restantes se inicializan de forma aleatoria y se entrenan con el dataset B.

Si el desempeño de $A3B$ es igual al de $B3B$, entonces se concluye que las primeras 3 capas son generales respecto a B. Si el desempeño se ve afectado, entonces las primeras 3 capas son específicas de A. El proceso se repite para valores de $n \in \{1, ..., 7\}$ en ambas direcciones. También se crearon otras dos redes alternativas:

- *Selffer network B3B*$^+$: igual a B3B pero en este caso no se congelan capas.
- *Transfer network A3B*$^+$: igual a A3B pero en este no se congelan capas.

Para entrenar y dado que el objetivo del paper no es obtener el mejor desempeño posible se utilizaron parámetros obtenidos desde otro paper. Cómo conclusiones los autores demuestran un método para cuantificar la capacidad de transferir features de cada capa de la red, revelando su generalidad o especificidad. Además demuestran que la transferencia se ve negativamente afectada por:

1. Dificultades en la optimización relacionadas al dividir redes en el medio, dado su fragilidad para adaptarse.
2. Especialización de las capas en la red originial a expensas del desempeño de la tarea objetivo.

Estos problemas dominan dependiendo si las *features* son transferidad al inicio, en el medio o al final de la red. Además se cuantifica que la brecha de transferencia crece a medida que la diferencia entre las tareas aumenta. Finalmente, se concluye que inicializar la red con *features* transferidas desde una tarea distinta es mejor que inicializar con pesos aleatorios.

2.2.2. Do Better ImageNet Models Transfer Better?

La investigación llevada a cabo (Kornblith et al., 2019) propone evaluar la relación entre arquitectura y *transfer learning*, fijando como hipótesis que aquellas arquitecturas que tienen mejor desempeño en el dataset *ImageNet* aprenden "mejores"*features* que pueden ser transferidas otras tareas de visión. Para testear dicho planteamiento comparan 16 redes convolucionales, entre ellas arquitecturas basadas en Inception, MobileNets, NASNets y DenseNets. Dichas redes fueron testeadas en 12 datasets de clasificación de imágenes, utilizando tres tipos de experimentos: entrenar redes desde cero (*random initialization*), hacer fine-tuned de la red pre-entrenada con *ImageNet* y utilizar la red como un *fixed feature extractor*. Para comparar entre diferentes dataset se utilizan transformaciones logaritmicas, dado que un incremento de un 1 % en un dataset es relativo a su *baseline*. Con esta transformación calculan la correlación entre el *accuracy* en *ImageNet* y el *accuracy* transferido a otros datasets.

Finalmente, se llega a la conclusión que existe una fuerte correlación entre el *accuracy* alcanzado en ImageNet y el obtenido al transferir este aprendizaje, así, el desempeño del modelo pre-entrenado es un factor crítico en *transfer learning*. Recalcando así la relevancia de utilizar modelos pre-entrenados para distintas tareas, lo que puede llevar a resultados en el estado del arte. Aspectos a considerar analizados por los autores fueron el tamaño del modelo, el número de parámetros y el tiempo de entrenamiento, concluyendo

que el tamaño del modelo y el número de parámetros no está correlacionado directamente con una mejor transferencia del aprendizaje, así tambien un tiempo de entrenamiento más largo no simepre conduce a una mejora significativa en la transferencia.

3. Descripción del modelo

Para poder construir nuestro modelo, se utilzó el *Dataset* de *Omnligot*, el cual presenta un conjunto de imágenes separados entre 50 alfabétos, cuyo lenguaje presenta entre 20 a 50 caracteres distintos. Cada uno de los caracteres posee 20 imágenes de muestra, cuyo formato cuenta con una dimensión de 105 x 105 píxeles.

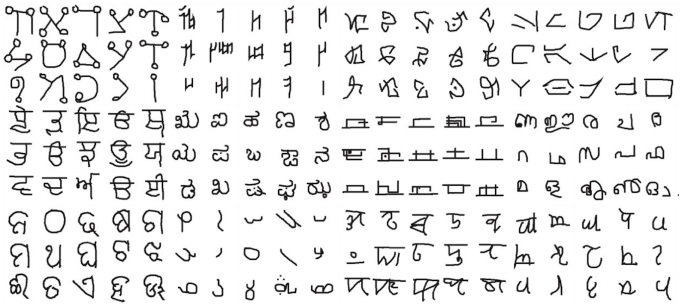


Figura 1: Ejemplos de caracteres del *Dataset Omniglot*

Para los experimentos, se tomó un subconjunto del *Dataset* llamado *images_background*, el cual presenta 30 alfabétos diferentes, cada uno con una cantidad determinada de caracteres. En total, se tomó una muestra de 964 caracteres, cada uno con 20 imágenes, lo cual arrojó un total de 19.280.

Luego, se realizó una separación del conjunto para tener *sets* de *training*, *validation* y *testing*. Para ello, 70 % de las muestras fueron para *training*, 20 % para *validation* y 10 % para *testing*, teniendo como consecuencia una proporción de 14, 4 y 2 imágenes de cada carácter respectivamenete.

Una vez teniendo los datos separados, se utilizaron diferentes modelos para entrenar el clasificador, entre ellos ***ResNet***, ***VGG16***, ***ResNetXt***, ***EfficientNet*** y ***RegNet***. Para la estimación de función de pérdida se utilizó *CrossEntropy*, y para el *learning rate* se usaron una variedad de valores con la finalidad de maximizar el *accuracy*. Dichos resultados se muestran en las tablas presentes en Resultados experimentales para cada uno de los clasificadores. En la siguiente figura se muestra el diagrama de bloques que representa la descripción realizada:

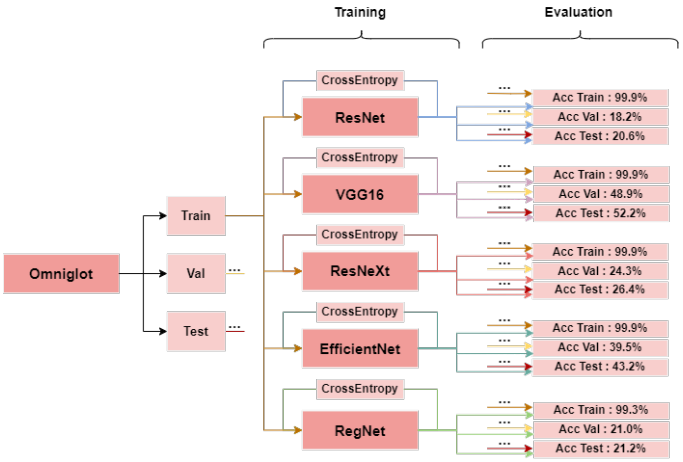


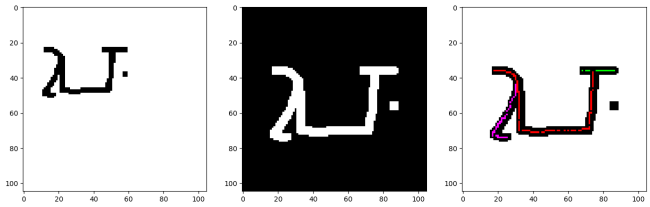
Figura 2: Diagrama de bloques del modelo de entrenamiento

Una vez obtenidos los modelos entrenados y evaluados, se procedió a elaborar el nuevo modelo que denominaremos ***VGGlot***, que tomará el mismo conjunto de imágenes y su distribución *train-validation-test* anterior para aplicarle una serie de cambios a través de *Transfer Learning Functions*.

La primera variación utilizada fue la de traslación de imágenes mediante *padding*. Para ello, cada uno de los caracteres fue desplazado 50 píxeles a la izquierda y 50 píxeles hacia arriba, de modo tal que el segmento del carácter quede posicionado en la esquina superior izquierda. El objetivo de esta variación será determinar cuánto influye la posición del carácter en la determinación de su respectiva clase.

La segunda variación utilizada fue la de aplicar un *background colorizer* a la imagen. Para ello, se procede a realizar una inversión en los colores, vale decir, si originalmente la imagen posee un fondo blanco y el carácter negro, entonces se aplica una transformación tal que quede el fondo negro y el carácter blanco. El objetivo de esta variación servirá para determinar cuánto influye el color y el fondo para la determinación de su respectiva clase.

La última variación utilizada fue la de utilizar el *stroke* de cada carácter. El *stroke* corresponde a los segmentos del carácter que son dibujados sin soltar el lapiz, de modo tal que cada segmento es resaltado con un color diferente dependiendo del orden en que fue escrito para formar el carácter completo. Esto nos servirá para observar cuánto influye el color del carácter para la determinación de su clase, respetando el fondo original.



(a) Carácter con *padding* (b) Carácter con *background* (c) Carácter con *Strokes*

Figura 3: Transformaciones utilizadas en los caracteres para VGGlot

A continuación se muestra el diagrama de bloques de la estructura de evaluación con los datos aumentados sobre el modelo VGGlot, utilizando VGG16 como modelo base y sus respectivos resultados.

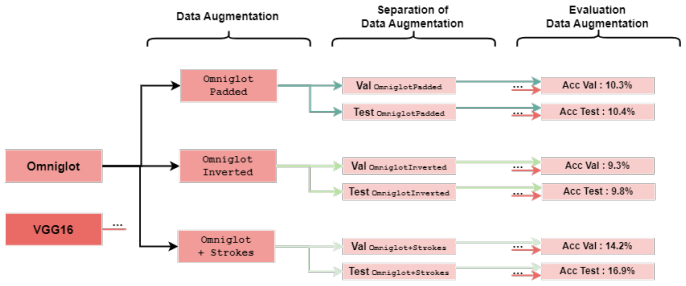


Figura 4: Diagrama de bloques de la evaluación del modelo VGGlot

Los diagramas y resultados para los otros modelos propuestos se encuentran en la sección de anexos.

4. Resultados experimentales

Con el fin de entrenar el mejor modelo que cumpliera con la tarea planteada sobre clasificación de caracteres de diversos alfabetos, realizamos varios experimentos con diversos modelos de aprendizaje profundo preentrenados. Entre estos modelos se obtuvieron los siguientes resultados:

4.1. Resultados VGG16

Modelo	Optimizador	Learning Rate	Params	Epoch	Acc Train	Acc Val	Acc Test
VGG16	SGD	5e-2	momentum=0.7	31	0.988	0.491	0.515
VGG16	SGD	5e-3	momentum=0.7	81	0.998	0.394	0.426
VGG16	Adam	1e-3	betas=(0.8, 0.9)	15	0.986	0.478	0.497
VGG16	Adam	1e-4	betas=(0.8, 0.9)	100	0.999	0.489	0.522
VGG16	Adam	1e-5	betas=(0.8, 0.9)	20	0.600	0.352	0.390

Tabla 2: Resultados del clasificador VGG16

4.2. Resultados EfficientNet

Modelo	Optimizador	Learning Rate	Params	Epoch	Acc Train	Acc Val	Acc Test
EfficientNet	Adam	1e-4	-	51	0.999	0.395	0.432
EfficientNet	Adam	8e-2	-	5	0.924	0.232	0.256
EfficientNet	Adam	2e-3	-	10	0.999	0.392	0.406
EfficientNet	Adam	8e-3	-	5	0.997	0.310	0.332
EfficientNet	Adam	1e-2	-	5	0.993	0.293	0.345
EfficientNet	Adam	4e-2	-	6	0.942	0.239	0.265
EfficientNet	Adam	4e-4	betas=(0.8, 0.9)	7	0.987	0.392	0.423
EfficientNet	Adagrad	9e-3	lr_decay=1e-4	100	0.999	0.383	0.423
EfficientNet	ASGD	0.01	10	0.49	0.156	0.179	

Tabla 3: Resultados del clasificador EfficientNet

4.3. Resultados ResNet

Modelo	Optimizador	Learning Rate	Params	Epoch	Acc Train	Acc Val	Acc Test
ResNet	SGD	1e-3	momentum=0.9	15	0.853	0.160	0.177
ResNet	SGD	1e-2	momentum=0.9	15	0.802	0.154	0.177
ResNet	SGD	1e-1	momentum=0.9	15	0.999	0.189	0.208
ResNet	SGD	1e-0	momentum=0.9	15	0.998	0.162	0.184
ResNet	Adam	1e-4	betas=(0.8, 0.9)	30	0.971	0.186	0.209
ResNet	Adam	1e-3	betas=(0.8, 0.9)	15	0.996	0.182	0.205

Tabla 4: Resultados del clasificador ResNet

4.4. Resultados ResNeXt

Modelo	Optimizador	Learning Rate	Params	Epoch	Acc Train	Acc Val	Acc Test
ResNeXt	SGD	1e-2	momentum=0.7	10	0.396	0.103	0.107
ResNeXt	SGD	5e-3	momentum=0.9	15	0.551	0.130	0.135
ResNeXt	SGD	5e-2	momentum=0.9	10	0.961	0.186	0.204
ResNeXt	SGD	1e-1	momentum=0.9	10	0.995	0.205	0.224
ResNeXt	SGD	1e-0	momentum=0.9	10	0.985	0.165	0.175
ResNeXt	SGD	5e-1	momentum=0.9	10	0.995	0.194	0.222
ResNeXt	SGD	5e-1	momentum=0.98	10	0.918	0.126	0.132
ResNeXt	SGD	5e-1	momentum=0.5	10	0.983	0.189	0.217
ResNeXt	Adam	1e-4	-	10	0.945	0.202	0.223
ResNeXt	Adam	1e-3	-	10	0.996	0.229	0.246
ResNeXt	Adam	1e-2	-	10	0.999	0.223	0.245
ResNeXt	Adam	1e-1	-	10	0.917	0.137	0.156
ResNeXt	Adam	5e-3	-	10	0.999	0.243	0.264
ResNeXt	Adam	3e-3	-	10	0.998	0.229	0.246
ResNeXt	Adam	7e-3	-	10	0.999	0.228	0.255
ResNeXt	Adam	6e-3	-	10	0.999	0.245	0.261

Tabla 5: Resultados del clasificador ResNeXt

4.5. Resultados RegNet

Modelo	Optimizador	Learning Rate	Params	Epoch	Acc Train	Acc Val	Acc Test
RegNet	SGD	1e-3	momentum=0.9	10	0.885	0.169	0.168
RegNet	SGD	1e-2	momentum=0.9	8	0.993	0.210	0.212
RegNet	SGD	1e-1	momentum=0.9	10	0.995	0.157	0.156
RegNet	Adam	3e-3	-	3	0.888	0.140	0.131

Tabla 6: Resultados del clasificador RegNet

A continuación se grafícan cinco de los mejores modelos y su desempeño en los sets de test y validación originales, así como también en los sets modificados:

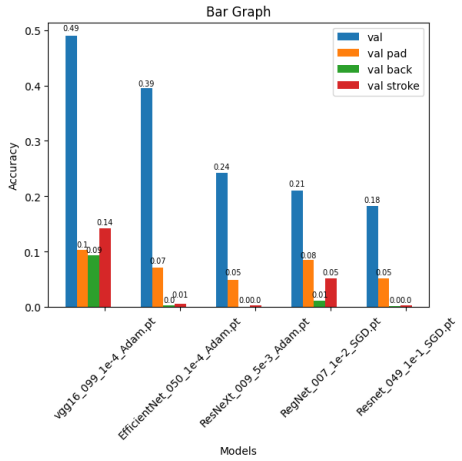


Figura 5: Resultados en sets de validación

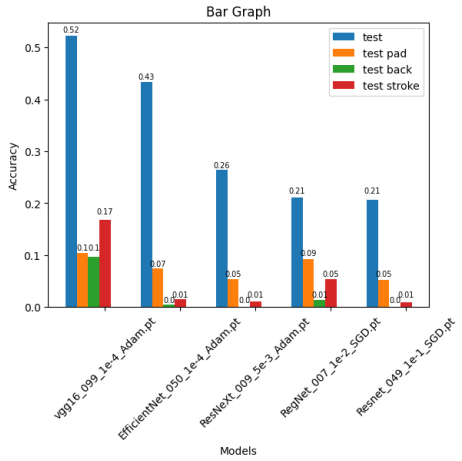


Figura 6: Resultados en sets de test

4.6. Análisis de Resultados

En base a los resultados obtenidos a lo largo del entrenamiento de cada uno de los clasificadores, se obtiene que el mejor desempeño logrado fue el del clasificador VGG16, con un *accuracy* del 51.5 % en el *set* de *testing*. Para ello se utilizó un *learning rate* de 0.0001 con un optimizador *Adam*. Por otra parte, el peor clasificador obtenido fue ResNeXt, el cual obtuvo un 10.7 % de *accuracy* usando un *learning rate* de 0.01 y optimizador *SGD*.

A partir de esto, se determina de que la tarea de clasificar el carácter entre las 964 clases diferentes resulta complicada para cada uno de los clasificadores, ya que ninguno pudo superar el 60 % de *accuracy*. Las razones de esta comportamiento se debe principalmente a la baja cantidad de imágenes por carácter que se utilizaron durante el entrenamiento.

Los modelos basados en NN tienden a ser los mecanismos más avanzados y óptimos para este tipo de problemas, sin embargo, suelen necesitar un gran volúmen de datos para poder converger correctamente, lo cual no fue este caso.

A pesar de estos resultados preliminares, si observamos los resultados de cada uno de estos modelos en el nuevo clasificador VGGlot, los *accuracy* para las imágenes alteradas mediante *Data Augmentation* bajaron considerablemente. Para el caso del mejor modelo preentrenado (VGG16) se obtuvo un *accuracy* de 10.4 % para los caracteres con *padding*. Para los caracteres con *inverted colors* se obtuvo 9.8 %, mientras que para los caracteres con *strokes* fue de 16 % de rendimiento.

Si observamos los gráficos presentados podemos ver que existe una tendencia en los desempeños. Claramente los datasets de validación y test originales poseen un mejor desempeño por sobre los otros datasets. Luego, en su mayoría los sets con *padding* obtienen el segundo mejor *accuracy*, exceptuando el modelo VGG16 que su segundo mejor desempeño son los datasets con *strokes*. En este caso podemos decir, que VGG16 le da menos importancia al color del carácter en sí y más a su forma, color del *background* y posición dentro de la imagen. En cambio, para las otras redes, el color del carácter es un factor crítico para determinar su clase, pero su posición o escala respecto de la imagen no lo es tanto. En todos los casos, los peores resultados son con los sets a los cuales se les cambia el color del *background* por lo que podemos terminar que para todas las redes esto es un factor importante. Notemos que los resultados obtenidos siguen la misma línea de lo presentado en el paper Geirhos et al., 2020, incluso se podría aventurar y afirmar que todas las redes están aprendiendo *shortcuts*, en este caso *features* que los humanos no consideramos al momento de escoger un carácter, como el color, la posición y el tamaño, en vez de prestarle especial atención a la forma o silueta de la figura.

Un aspecto importante a notar es que, en este caso, el modelo VGG16 resultar ser el de mayor tiempo de ejecución, incluso en la evaluación. Probablemente esto se deba a los parámetros utilizados y la manera en que se van adaptando a medida de que se entrena el modelo, como lo es el caso del número de épocas y el *learning rate* utilizado.

Ante estos resultados, claramente es posible evidenciar una notoria baja en el *accuracy*, pues hubo una disminución entre 3 a 5 veces el *accuracy* obtenido para el modelo sin *Data Augmentation*. Esto sin lugar a dudas son resultados no deseados, pues implica que los modelos no son capaces de mantener el rendimiento al enfrentarse a los carácteres con ligeras modificaciones pero que conservan la misma estructura. ¿Implica esto que el modelo generado es deficiente para la generalización? Es difícil argumentar por qué los modelos

serían buenos para este tipo de problemas, teniendo como datos los resultados que se obtienen en este experimento. Sin lugar a dudas el modelo está estimando ciertos patrones y *shortcuts* al momento de generalizar. Sin embargo, toman en cuenta valores como color, posición y tamaño, los cuales no necesariamente son los aspectos más relevantes para determinar si un carácter corresponde o no a una determinada clase. Debido a esto, no se pudo lograr el objetivo de que el modelo prestará mayor atención a la forma o silueta de la figura, el cual es el factor más determinante o más bien la manera en que los humanos tienden a observar en las figuras.

5. Conclusiones

A modo de conclusión, se puede decir que el nuevo modelo VGGlot planteado es capaz de identificar ciertos patrones y *shortcuts* para evaluar o reconocer una imagen, como lo son el color, posición y tamaño, lo cual se ve reflejado en el *accuracy* obtenido en la Figura 4. Sin embargo, era de esperar que los resultados de este modelo fueran similares a los obtenidos por medio del clasificador entrenado. Lamentablemente, hubo mucha diferencia, ya que el rendimiento del modelo VGGlot fue entre un 10 % a un 20 % dependiendo de la transformación, mientras que el del clasificador VGG16 tuvo un *accuracy* por sobre el 50 %. En base a esto, no se logró el objetivo de que el modelo pudiera considerar factores más relevantes al momento de ajustar sus parámetros, como lo son la forma de la silueta de los carácteres, similar al comportamiento humano.

Cabe destacar que el *Dataset Omniglot* utilizado pudo haber influido fuertemente en estos análisis, pues el reconocimiento de caracteres suele ser una tarea compleja debido a que el vector de píxeles tomado es bastante similar entre cada una de las muestras. En otro tipo de tareas de reconocimiento, como lo es el ámbito humano-facial, puede que los resultados sean bastante más altos, por lo que quedaría como evaluación futura probar este mismo mecanismo para otros conjuntos de datos diferentes al actual.

Como trabajo futuro, se podrían investigar más modelos e ir alterando los hiperparámetros para que el rendimiento inicial de cada uno de los modelos sea por sobre el 90 % para la tarea de reconocimiento del carácter. Posterior a ello, evaluar el modelo VGGlot con *Data Augmentation* y ver si el análisis de resultados se sigue conservando. Por otra parte, también es posible investigar de nuevos mecanismos de entrenamiento, como lo es el uso de ensambladores. En dichos casos, se puede concatenar el resultado del entrenamiento de diferentes modelos para tener un modelo más robusto y general que sea capaz de generalizar mejor ante distintos escenarios.

6. Anexos

6.1. Modelos y Resultados de clasificadores

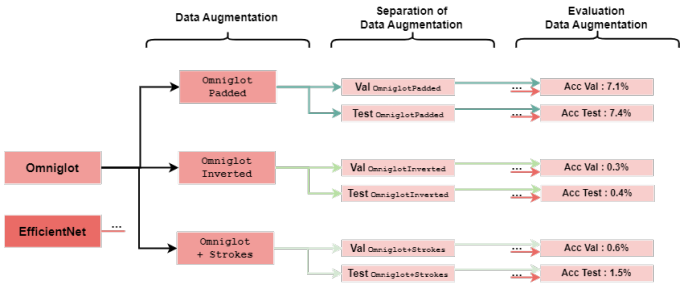


Figura 7: Diagrama de bloques del modelo VGGlot con EfficientNet

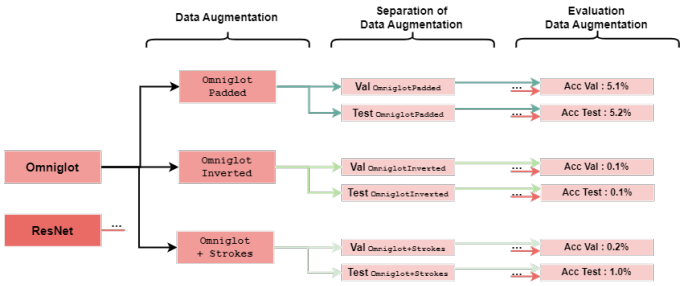


Figura 8: Diagrama de bloques del modelo VGGlot con ResNet

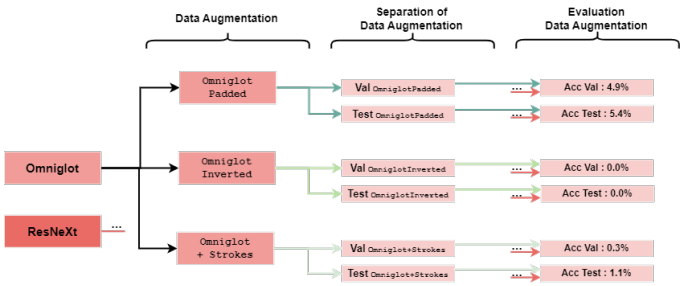


Figura 9: Diagrama de bloques del modelo VGGlot con ResNeXt

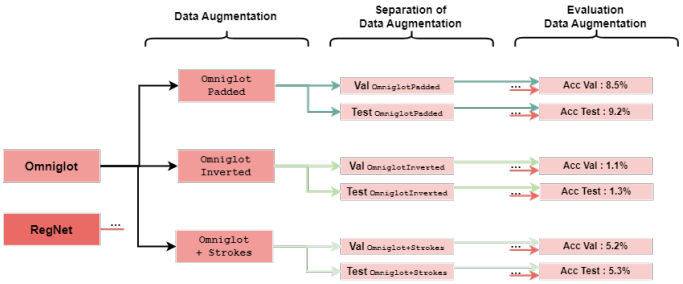


Figura 10: Diagrama de bloques del modelo VGGlot con RegNet

Los resultados al evaluar los datos de validación y testeo del *dataset* original y sus 3 variaciones aumentadas, sobre múltiples pesos de los 5 modelos propuestos.

model	val	test	val pad	test pad	val back	test back	val stroke	test stroke
vgg16_099_1e-4_Adam.pt	0.489367	0.522303	0.103216	0.104253	0.0931017	0.0975104	0.141857	0.168568
vgg16_080_5e-3_SGD.pt	0.393932	0.426349	0.094139	0.0850622	0.0736515	0.0819502	0.116442	0.155083
vgg16_030_5e-2_SGD.pt	0.491183	0.515042	0.100882	0.10166	0.0915456	0.0928423	0.139782	0.174793
vgg16_015_1e-3_Adam.pt	0.477956	0.497407	0.103994	0.104253	0.0969917	0.0975104	0.136929	0.161826
EfficientNet_009_2e-3_Adam.pt	0.389264	0.406639	0.0746888	0.0845436	0.00389004	0.00363071	0.00440871	0.0134855
EfficientNet_099_1e-2_ASGD.pt	0.381224	0.422199	0.0705394	0.0855809	0.00337137	0.00259336	0.00596473	0.0150415
EfficientNet_050_1e-4_Adam.pt	0.39471	0.43361	0.0713174	0.0741701	0.00337137	0.00414938	0.0064834	0.0145228
ResNeXt_009_1e-3_Adam.pt	0.228734	0.246369	0.0459025	0.0446058	0.00103734	0.00103734	0.00363071	0.00778008
ResNeXt_009_1e-2_Adam.pt	0.22277	0.244813	0.0516079	0.0544606	0.00103734	0.00103734	0.00492739	0.0129668
ResNeXt_009_5e-3_Adam.pt	0.242739	0.264004	0.0490145	0.0544606	0.000259336	0.000518672	0.00363071	0.0108921
ResNeXt_009_6e-3_Adam.pt	0.245332	0.260892	0.0484959	0.0544606	0.00103734	0.00103734	0.00440871	0.0108921
RegNet_009_1e-1_SGD.pt	0.145488	0.149896	0.0606847	0.0653527	0.0064834	0.0103734	0.0337137	0.0342324
RegNet_007_1e-2_SGD.pt	0.210062	0.211618	0.0845436	0.0923237	0.0108921	0.0134855	0.0518672	0.0534232
resnet50_049.pt	0.178164	0.192427	0.0500519	0.0487552	0.00103734	0.00103734	0.00414938	0.00674274
Resnet_049_1e-0_SGD.pt	0.171421	0.185166	0.0422718	0.0456432	0.00103734	0.00155602	0.00337137	0.0103734
Resnet_049_1e-1_SGD.pt	0.182054	0.205913	0.0510892	0.0529046	0.00103734	0.00103734	0.00259336	0.00985477
Resnet_049_1e-2_SGD.pt	0.182313	0.19917	0.0526452	0.0508299	0.00181535	0.00103734	0.00233402	0.00726141

Tabla 7: Resultados de diversas versiones de los clasificadores con datos aumentados

Si se quieren recrear los resultados o probar los modelos, todo esto se encuentra en nuestro repositorio **VGGlot**.

Referencias

Chan, S. C., Dasgupta, I., Kim, J., Kumaran, D., Lampinen, A. K., & Hill, F. (2022). Transformers generalize differently from information stored in context vs in weights. arXiv preprint arXiv:2210.05675.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. Nature Machine Intelligence, 2(11), 665-673.

Kornblith, S., Shlens, J., & Le, Q. V. (2019). Do better imagenet models transfer better? Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2661-2671.

Lake, B., & Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. International conference on machine learning, 2873-2882.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? Advances in neural information processing systems, 27.