Tarea 4

MA5203 – Aprendizaje de Máquinas Probabilístico

|  |  |
| --- | --- |
| Nombre: | Sebastián Parra |
| Profesor: | Felipe Tobar |
| Auxiliar: | Alejandro Cuevas |
|  | Cristóbal Valenzuela |
|  | Lerko Araya |
|  | Mauricio Campos |
|  | Nicolás Atamayo |
| Fecha: | 2 de julio de 2018 |

# **Parte 1: Redes Neuronales y MNIST**

## **Arquitectura de la Red**

En la tabla 1 se muestran las especificaciones de las capas que formaron parte de la red convolucional utilizada en la tarea.

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Tipo | Tamaño de Filtro | *Padding* |
| conv1 | Convolución | 28 canales de 2x2 | Same |
| pool1 | Max Pooling | 2x2 | Same |
| conv2 | Convolución | 28 canales de 2x2 | Same |
| pool2 | Max Pooling | 2x2 | Same |
| fc1 | Fully Connected | 196 unidades | N/A |
| cl\_layer | Clasificación | 10 unidades | N/A |

# **Tabla 1:** Arquitectura de la red convoucional

## **Ajuste de Hiperparámetros y Optimizador elegido**

En la tabla 2 se muestran los hiperparámetros y el algoritmo de optimización utilizado, junto con detalles sobre su implementación en esta tarea

|  |  |  |
| --- | --- | --- |
| Parámetro | Valor | Detalles |
| Función de Activación | ReLU, Max, Argmax | ReLU para capas convolucionales, Max para capas de pooling, y Argmax para la capa de clasificación |
| Función de Costo | Entropía Cruzada con Softmax | N/A |
| Tamaño de Batch | 40 | N/A |
| Optimizador | Adam | Parámetros por defecto de tensorflow (tasa de aprendizaje 0.001, beta1 0.9, beta2 0.999, epsilon 1e-08) |

**Tabla 2:** Hiperparámetros y optimizador de la red convolucional

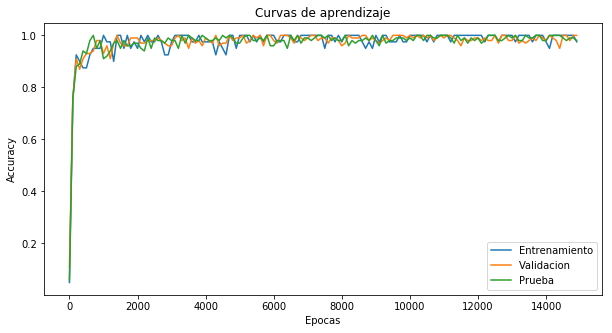
## **Regularización y Entrenamiento**

Como método de regularización se aplicó la técnica de *dropout* sobre la capa *fully connected* de la red, utilizando una probabilidad del 50% para eliminar una unidad.

Para el entrenamiento de la red, se aplicó el algoritmo de *backpropagation* con los parámetros anteriormente mencionado por 15000 épocas, donde además por cada 100 iteraciones se registraron las tasas de acierto en los conjuntos de entrenamiento, validación, y prueba. Cabe mencionar que en esta sección se tuvo problemas con evaluar los conjuntos de validación y prueba en su totalidad debido a que el programa se quedaba sin memoria disponible (al tener que lidiar con tensores de dimensiones 5000x28x28x28 y 10000x28x28x28 respectivamente), por lo que para obtener estas métricas se utilizaron batches de 100 imágenes de cada conjunto.

## **Resultados**

Para evaluar el desempeño de la arquitectura, se estudiaron dos métricas: La curva de aprendizaje para los conjuntos de entrenamiento, validación y prueba (figura 1), y la matriz de confusión sobre este último conjunto (tabla 3). Debido a los problemas de memoria mencionados en la sección anterior, la matriz de confusión se calculó utilizando 1000 imágenes del conjunto total de prueba.



**Figura 1:** Curvas de aprendizaje (Tasa de acierto versus número de épocas) para los conjuntos de entrenamiento, validación y prueba

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **0** | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0.01 | 0.98 | 0 | 0 | 0 | 0 | 0.01 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0.99 | 0 | 0 | 0 | 0 | 0.01 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0.99 | 0 | 0 | 0 | 0 | 0.01 |
| **5** | 0 | 0 | 0 | 0.01 | 0 | 0.97 | 0 | 0.01 | 0 | 0.01 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0.01 | 0.98 | 0 | 0.01 | 0 |
| **7** | 0 | 0.01 | 0.01 | 0 | 0 | 0 | 0 | 0.98 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 0 | 0.01 | 0 | 0 | 0.99 | 0 |
| **9** | 0 | 0 | 0 | 0 | 0.01 | 0 | 0 | 0.02 | 0 | 0.97 |

**Tabla 3:** Matriz de confusión por clase.

## **Análisis de Resultados y Conclusiones**

Lo primero que se puede notar en las curvas de aprendizaje de la figura 1 es el evidente ruido en cada una de las líneas, explicado por la utilización de batches en la evaluación, que agregan aleatoriedad. Además, se puede apreciar que la red neuronal aprende rápidamente en las primeras épocas y luego, a partir de la iteración 4000 en adelante aproximadamente, pareciera no aprender nada relevante, ya que las variaciones en las tasas de acierto pasan a ser predominantemente por ruido estocástico.

En cuanto a la matriz de confusión, se puede notar que ésta presenta una alta tasa de acierto para cada clase, por lo cual se puede concluir que la red convolucional logra tener un buen desempeño en este problema. Además, se puede notar que la mayoría de las malas clasificaciones son con números parecidos (por ejemplo, el 1 con el 7, el 2 con el 7, el 4 con el 9, etc.)

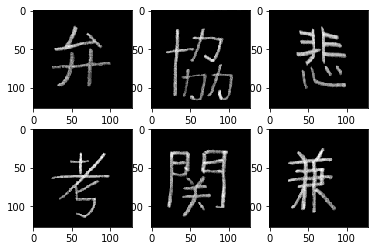
En resumen, a través de esta tarea se demuestra que las redes convolucionales son una poderosa y relativamente fácil de implementar (no es necesario realizar un ajuste complejo para obtener buenos resultados) herramienta para resolver problemas de reconocimiento de imágenes cuya principal limitación es la cantidad de recursos computacionales que puede consumir.

# **Paso 2: Proyecto**

## **Clasificación de kanjis manuscritos**

El problema consiste en desarrollar un clasificador que permita distinguir distintos caracteres de kanji (uno de los alfabetos japoneses caracterizado por su alta complejidad y gran similitud entre sus caracteres) a partir de imágenes con símbolos manuscritos (similar al caso de MNIST). La base de datos a utilizar se conoce como ETL-8 y se puede encontrar en <http://etlcdb.db.aist.go.jp/>. Ésta consiste en 140480 imágenes de 878 diferentes kanjis más 12.000 imágenes de 75 letras distintas del alfabeto hiragana. Cada imagen es de 64 x 64 píxeles.

A través de la inspección de la base de datos realizada en la tare anterior, se dedujo que la porción de la base de datos a utilizar consistía de 960 imágenes en total, con 160 imágenes por clase, por lo que el problema presenta 6 clases distintas, las cuales se muestran en la figura 2.



**Figura 2:** Clases de la base de datos

Debido a los buenos resultados obtenidos en la parte 1 de esta tarea y dado que el proyecto se trata de un problema de clasificación de imágenes, se pretende utilizar una red convolucional para su resolución. Sin embargo, dado que se requiere realizar un estudio extenso de *fine-tuning* para obtener los mejores resultados posibles, se decide utilizar primero un clasificador *Random Forest*, caracterizado por su buen desempeño y muy rápida implementación, para obtener un valor de certeza que servirá como un piso mínimo al analizar el desempeño de la red neuronal. En el archivo *proyecto.ipynb* se encuentra un código donde se carga y preprocesa la base de datos, se visualizan las clases del problema, y se entrena un modelo rápido de random forest, obteniendo una tasa de acierto del 92% para el conjunto de prueba aproximadamente.