**Prácticas Profesionalizantes – Rotación 2**

**Tema: Redes Neuronales**

**Bitácora Personal**

**Michel Leandro**

**Sede De Práctica Profesionalizante: UTN - Delta - Campana**

**Responsable a Cargo: Cerrotta Santiago**

**Alumno: Leandro Michel**

**DESARROLLO**

***16/08/2024***

Siendo esta la continuación del primer cuatrimestre de esta práctica profesionalizante tuve como primera tarea volver a revisar el trabajo hecho para refrescar la mente además de organizar y preparar mi entorno de trabajo instalando software o librerías necesarias.

***23/08/2024***

Fijamos los objetivos de esta segunda etapa, los cuales en un principio serían profundizar las redes, mejorar códigos, bajar cantidad de parámetros, poder sacar una conclusión.

Luego aplicar redes pre-entrenadas, aprender de las mismas, segmentar nanopartículas y obtener datos de sus tamaños para finalmente intentar hacer una CNN que pueda encontrar fallas en tableros para una empresa.

***26/08/2024***

Buscando una mejor arquitectura que la definida en la primera etapa pero con la menor cantidad de parámetros posible.

**ARQUITECTURA DEFINIDA EN LA PRIMERA ETAPA**

|  |  |
| --- | --- |
| conv2d, 32f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d 64f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 1024n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 5  BatchSize: 32  Time x Epoch: 78s  Total time: 6.5min  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 95,00%  Params: 367,043,541  Dense 512n: 183,530,964 |

**PRUEBAS RGB MAÍCES SOLOS**

|  |  |
| --- | --- |
| conv2d, 16f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d 32f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 512n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 24s  Total time:  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 91.50%  Params: 91,766,260 |

|  |  |
| --- | --- |
| conv2d, 8f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d 16f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 512n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 11s  Total time:  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 93.50%  15 Epoch: 89.50%  Params: 45,887,364 |

|  |  |
| --- | --- |
| conv2d, 16f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 512n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 32s  Total time:  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 95.00%  Params: 183,512,020  Conv2 8f: 91,761,396 - 93% |

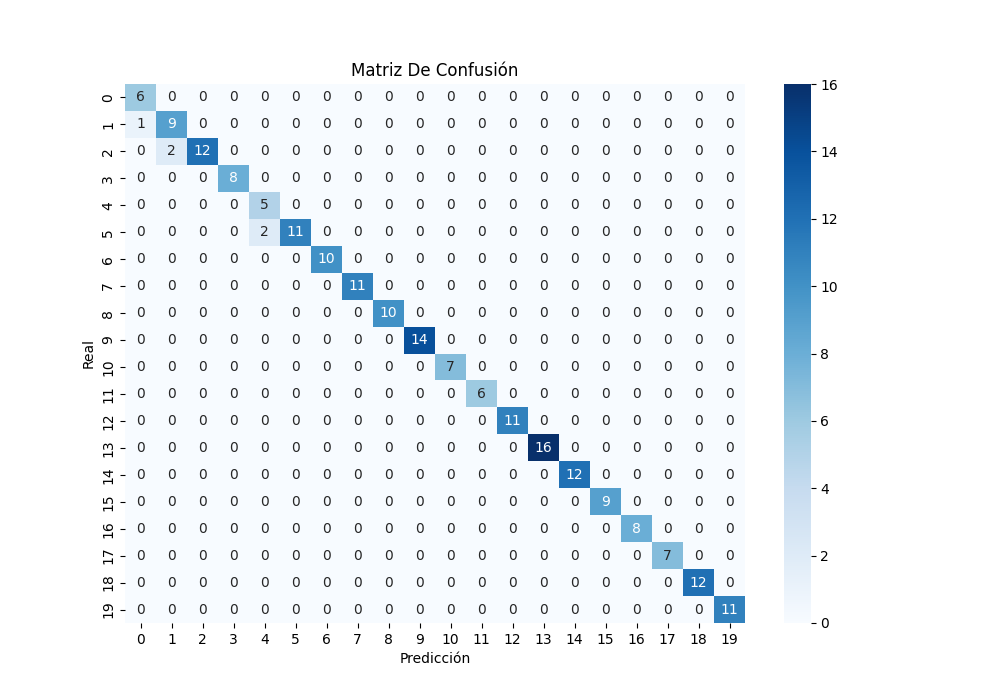
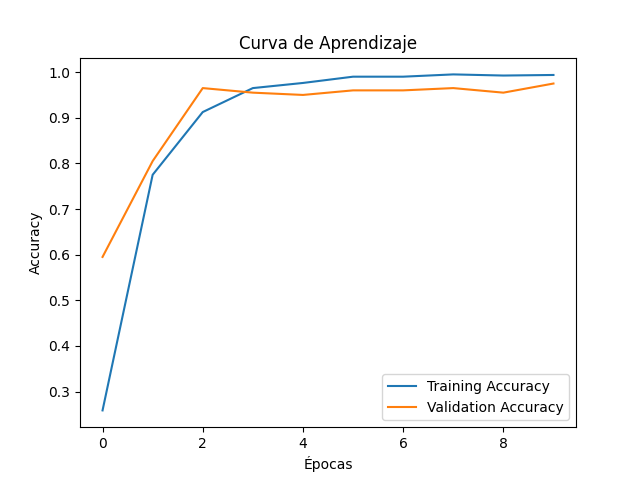
|  |  |
| --- | --- |
| conv2d, 8f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d, 16f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d, 32f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 512n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 22s  Total time:  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 87.50%  Params: 22,954,404 |

|  |  |
| --- | --- |
| conv2d, 4f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d, 8f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 512n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 6s  Total time:  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 95.50%  Params: 22,948,780  Agregando otra Conv2 16f  11,481,148 - 91%  Poniendo Dense 256n  11,474,604 - 95% |

|  |  |
| --- | --- |
| conv2d, 2f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d, 4f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 512n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 3s  Total time:  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 93.50%  Params: 11,479,704  Agregando otra Conv2 8f  5,745,600 - 90% |

**MEJOR ARQUITECTURA ENCONTRADA EN RGB MAÍCES SOLOS**

|  |  |
| --- | --- |
| conv2d, 2f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d, 4f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 256n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 3s  Total time:  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 97.50%  Params: 5,739,928  Grises: 97% |

****

***28/08/2024***

A partir de la mejor arquitectura lograda intento encontrar algo mejor cambiando parámetros más específicos (Dropout, Tamaño de filtro, Pooling, Funcion de activacion).

**Conv2d 2f Desde 4x4 Hasta 10x10 = 94% Promedio (5x5=96%)**

**Conv2d 4f Desde 4x4 Hasta 10x10 = 89% Promedio (5x5=34%)**

**Sin 1er Dropout = 95% - Sin 2do Dropout = 82% - Sin 3er Dropout = 94%**

**Sin Ningún Dropout = 95%**

**1er Pooling 3x3-4x4-5x5 = 90% - 2do Pooling 3x3-4x4-5x5 = 97%**

**Funcion De Activacion De Capa De Salida ‘Sigmoid’ = 95%**

**Funcion De Activacion De Capa De Salida ‘Linear’ = 5%**

**Conv2d 2f Stride 2 Hasta 5 = 87% Promedio - Conv2d 4f Stride 2 Hasta 5 = 86% Promedio**

**1er Pooling Stride 3 Hasta 5 = 90% - 2do Pooling Stride 3 Hasta 5 = 91%**

|  |  |
| --- | --- |
| conv2d, 2f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d, 4f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 256n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 3s  Total time:  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 97.50%  Params: 5,739,928  Grises: 97% - 5,739,892 - 65MB  Maíces-Lentejas: 93% |

**Fotos En Las Que Se Equivoca**



***02/09/2024***

**PRUEBAS RGB MAÍCES LENTEJAS**

**Multiplicando x2 La Cantidad De Filtros De Ambas Capas Hasta 32f-64f = 92%**

**Conv2d 2f Desde 4x4 Hasta 10x10 = 93% Promedio (7x7=96% Params: 5,740,168)**

**Conv2d 4f Desde 4x4 Hasta 10x10 = 75% Promedio**

**1er Pooling 3x3-4x4-5x5 = 88% - 2do Pooling 3x3-4x4-5x5 = 90% (3x3 = 96.5%)**

**Sin 1er Dropout = 90% - Sin 2do Dropout = 79% - Sin 3er Dropout = 79%**

**Sin Ningún Dropout = 91%**

**Dense 512n = 96% - Dense 1024n = 95%**

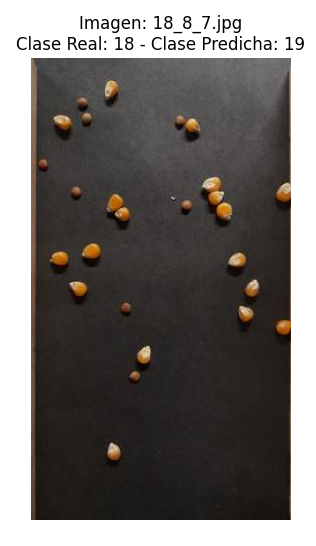
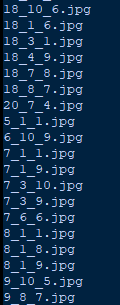
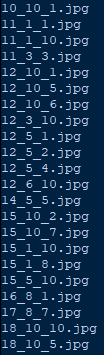
**Funcion De Activacion De Capa De Salida ‘Sigmoid’ = 87%**

**Funcion De Activacion De Capa De Salida ‘Linear’ = 15%**

**Conv2d 2f Stride 2 Hasta 5 = 75% Promedio - Conv2d 4f Stride 2 Hasta 5 = 86% Promedio**

**1er Pooling Stride 3 Hasta 5 = 87% - 2do Pooling Stride 3 Hasta 5 = 87%**

|  |  |
| --- | --- |
| conv2d, 2f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d, 4f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 256n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 3s  Total time:  num Imagenes: 2000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 93%  Params: 5,739,928 |

****

**MORALEJA: Después de todas las pruebas realizadas para las imágenes de maíces con lentejas se observa que la red óptima para solo maíces funciona también con gran precisión para estas imágenes. Se observan leves mejoras al aumentar la cantidad de neuronas de la capa de clasificación, el tamaño del pooling de la segunda capa a 3x3 y el tamaño del filtro de la primera capa.**

**Conv2d 2f 7x7 - 2do Pooling 3x3 - Dense 512n = 91%**

**Conv2d 2f 7x7 - 2do Pooling 3x3 - Dense 256n = 90.5%**

**Conv2d 2f 7x7 - 2do Pooling 2x2 - Dense 256n = 94%**

**Conv2d 2f 7x7 - 2do Pooling 2x2 - Dense 512n = 95.5%**

**Conv2d 2f 3x3 - 2do Pooling 3x3 - Dense 512n = 91.5%**

**Conv2d 2f 3x3 - 2do Pooling 3x3 - Dense 256n = 86%**

**Conv2d 2f 3x3 - 2do Pooling 2x2 - Dense 512n = 91%**

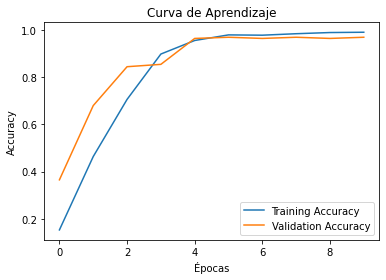
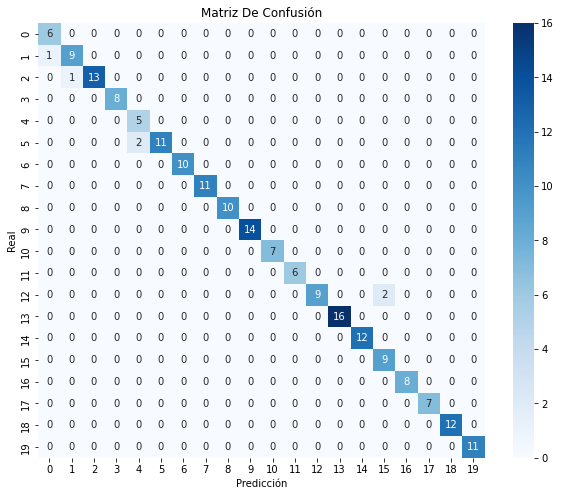
**MORALEJA: Usar Conv2d 2f 7x7 solo nos aumenta 1000 parámetros y nos da una muy buena precisión en comparación a otros casos, por otro lado el 2do Pooling en 3x3 no sería una buena opción ya que en la mayoría de casos solamente bajo la precisión.**

Curva con dense 512n - Conv 2d 2f 7x7 - Pooling 2x2 de maíces con lentejas - Params: **11,479,944** - **97%**

Usando esa arquitectura con maíces solos: **96.5%**

256n Params: **5,740,168** - 2d Pooling 3x3 Params: **5,012,360**

**MORALEJA: Al duplicar la cantidad de neuronas de la última capa se duplican los parámetros. Y su vez si aumento si aumento el pooling a 3x3 en vez de 2x2 puedo disminuir la cantidad de parámetros a la mitad.**

**PROMEDIOS**

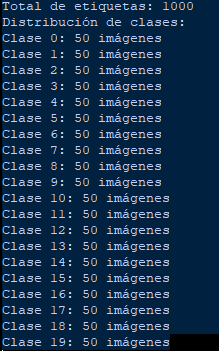
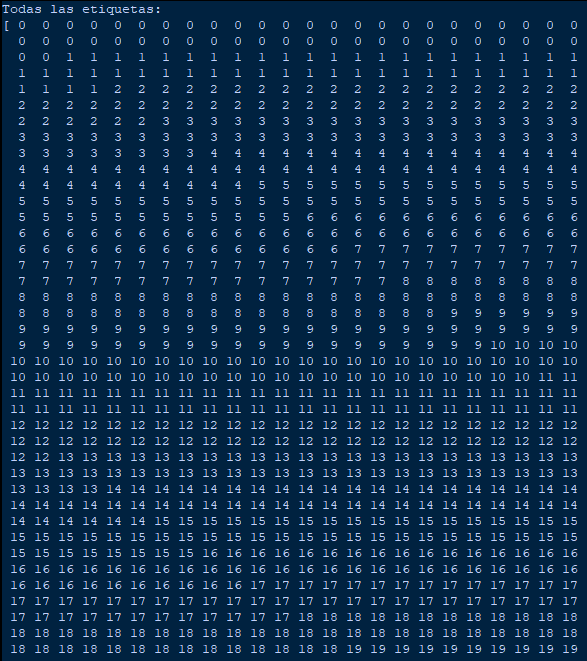
**Maíces Solos Mejor Arq: 92.5% – 95% – 87% – 92% - 96.5% – 92% – 90% – 91% – 94% – 90% = 92% MSE: 0.5200**

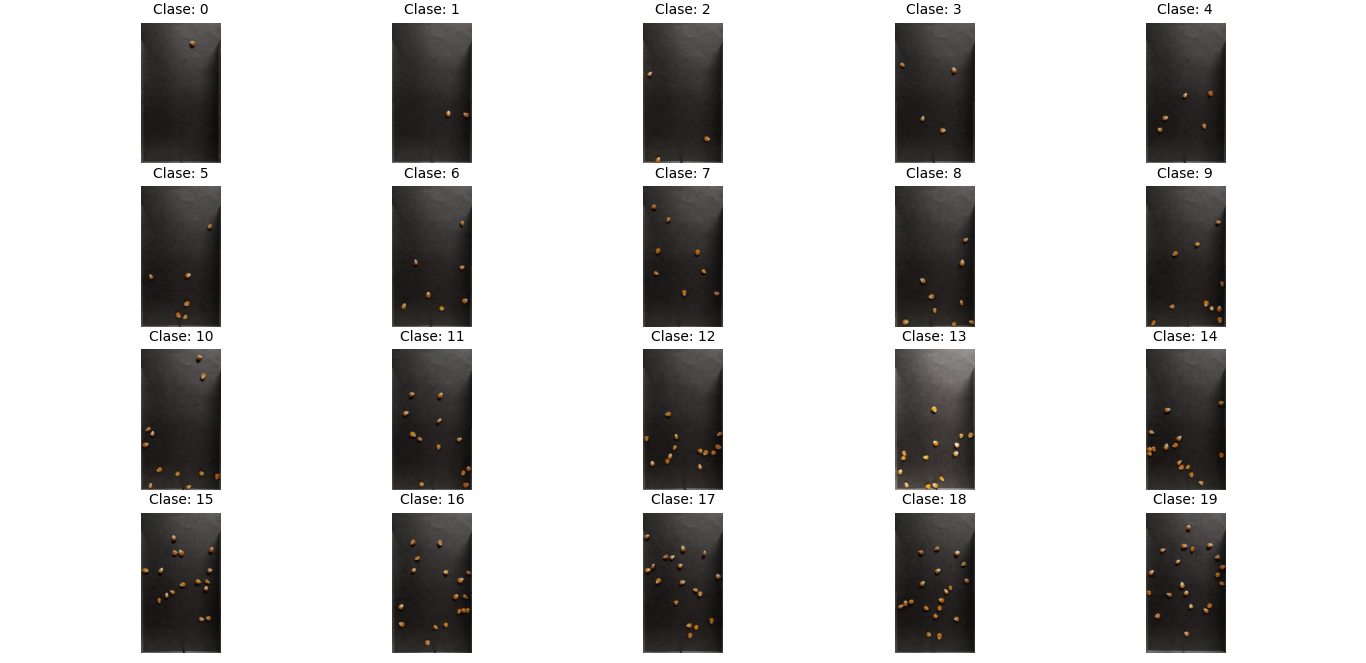
**Maíces Solos Con Mejor Arq De Maíces Lentejas: 93,5% - 92,5% - 94% - 91,5% - 94% - 94,5% - 92% - 94% - 94,5% - 93,5% = 93,5% - MSE: 0.4100**

**Maíces Lentejas Mejor Arq: 84.25% - 86.25% - 81.75% - 89.00% - 83.75% - 84.50% - 82.25% - 84.50% - 89.50% - 82.75% = 84,95% - MSE: 1.1430**

**MORALEJA: Los entrenamientos tienen mucha variabilidad. Recordar que para caracterizar una red hay que hacer al menos 5 entrenamientos para saber la variabilidad de la arquitectura y tener un promedio.**

Correcciones en orden de labels y images.

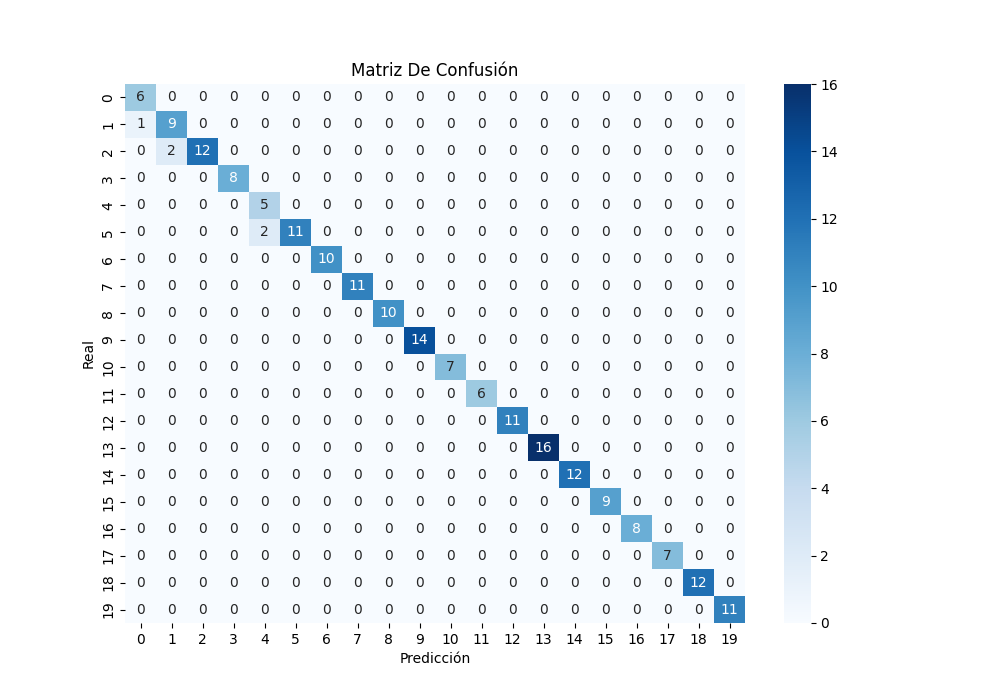
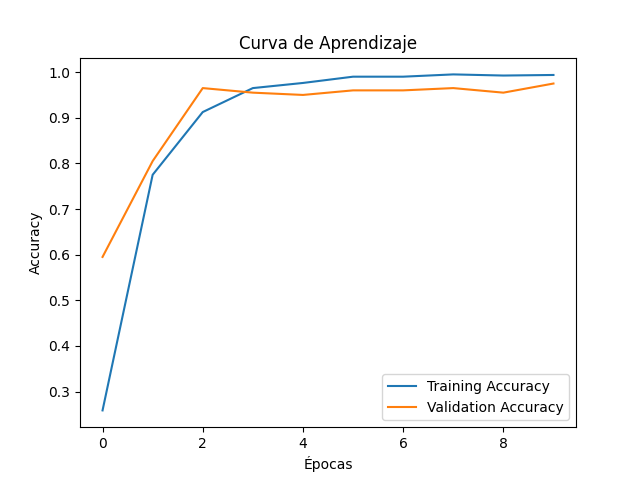


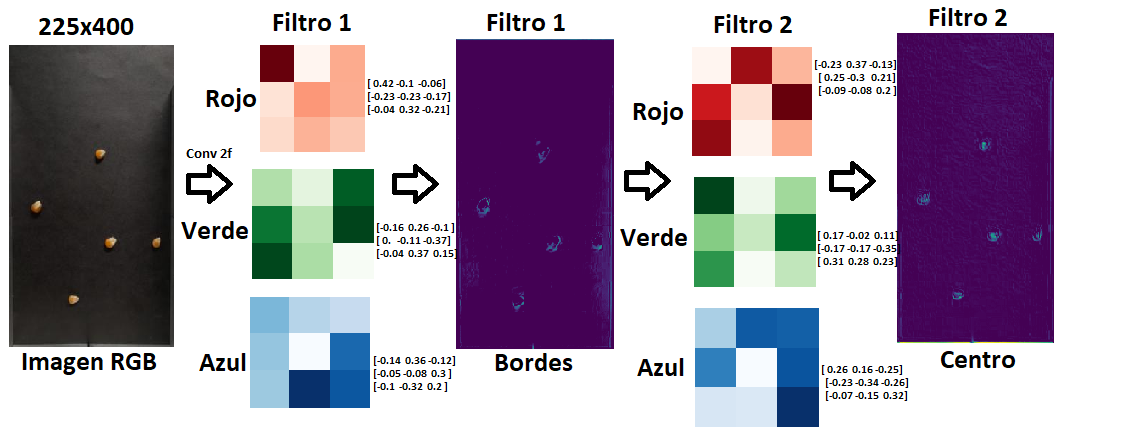


***17/09/2024***

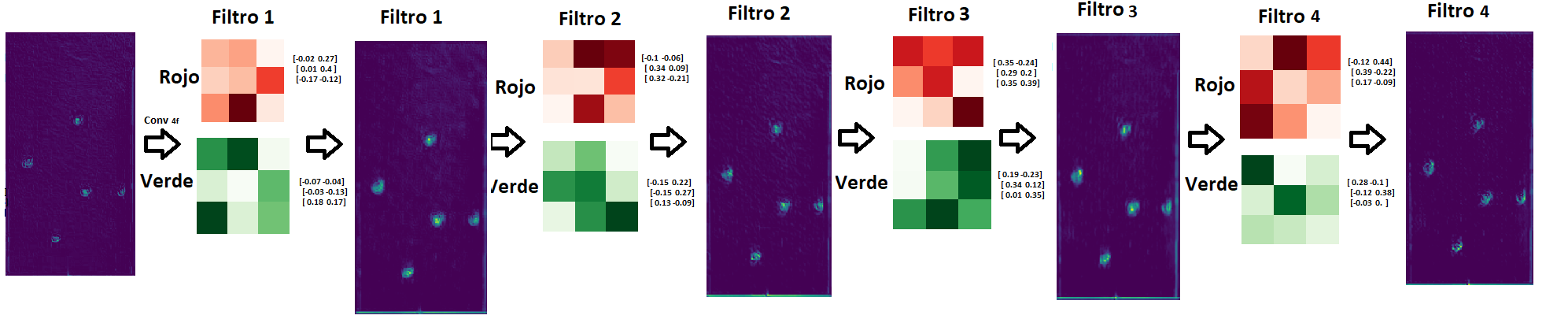
**ARQUITECTURAS DEFINIDAS**

|  |  |
| --- | --- |
| conv2d, 2f, 3x3, relu  maxPooling 2x2  Dropout 0.25  conv2d, 4f, 3x3, relu  maxPooling 2x2  Dropout 0.25  Flatten  Dense 256n, relu  Dropout 0.15  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 3s  MAÍCES SOLOS  num Imagenes: 1000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 97.50%  Params: 5,739,928 |

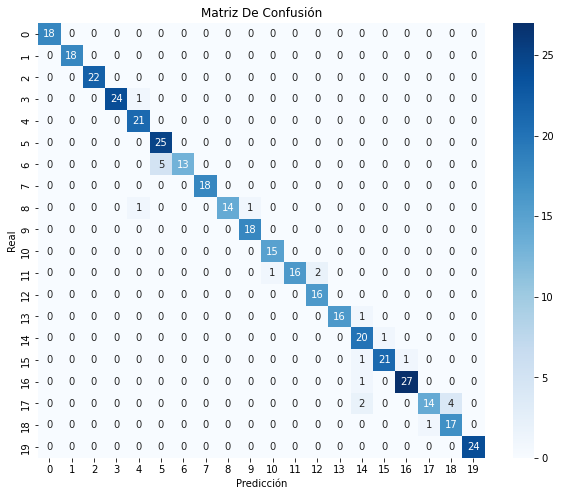
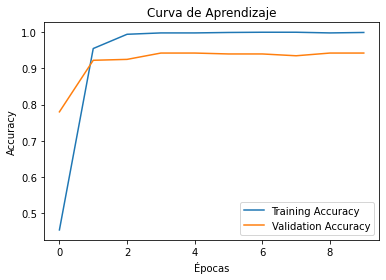
****

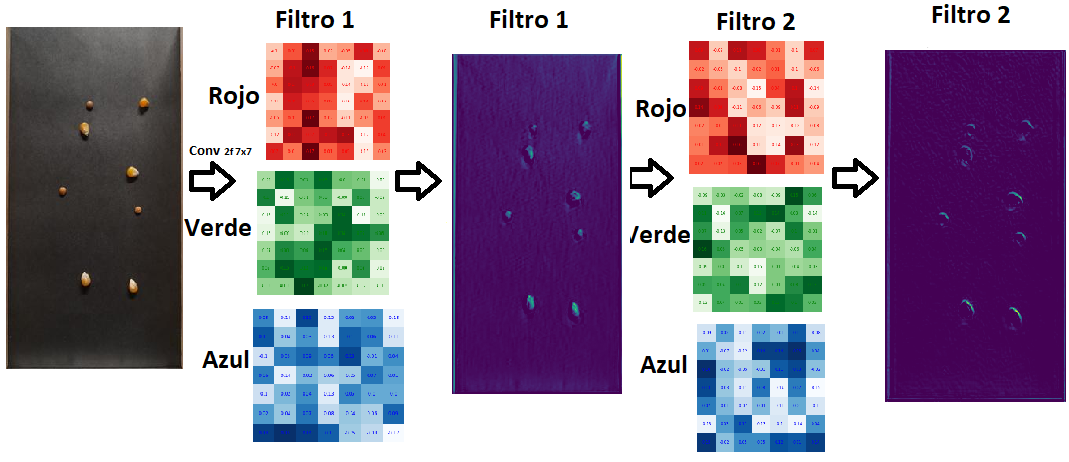
****

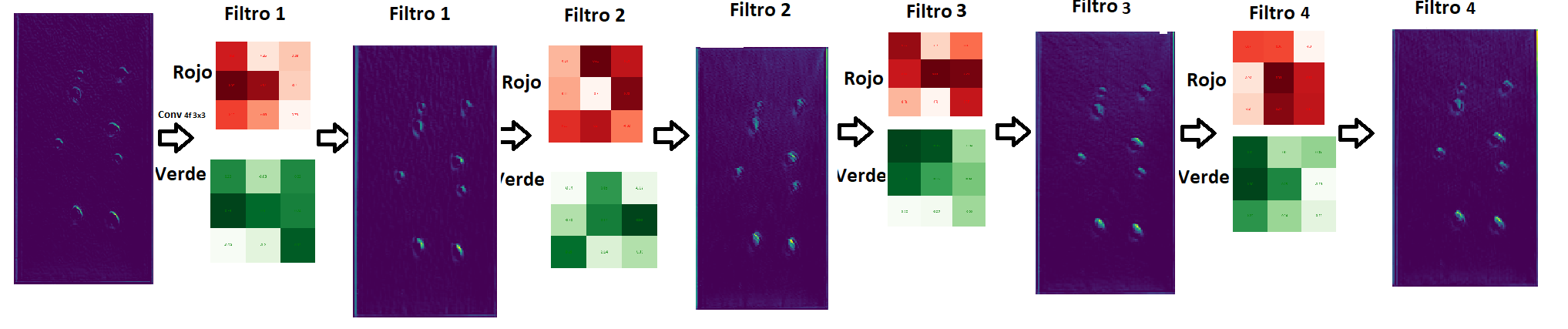
**MORALEJA:Se puede apreciar como el filtro uno detecta bordes o sombras a diferencia del filtro dos que se enfoca más en los centros de las formas que encontró, en las siguientes capas empieza a profundizar más sobre las características de dichas formas.**

****

|  |  |
| --- | --- |
| conv2d, 2f, 7x7, relu  maxPooling 2x2  conv2d, 4f, 3x3, relu  maxPooling 2x2  Flatten  Dense 512n, relu  Dense (numClases)n softmax | Epochs: 10  BatchSize: 32  Time x Epoch: 9s  MAÍCES LENTEJAS  num Imagenes: 2000 RGB  resolución: 225alto x 400ancho  num Clases: 20 (50 fts x 20 class)  testAccuracy: 94.25%  Params: 11,479,944 |







También realice un código que devuelve un excel con los outputs de la capa flatten con los valores del vector bidimensional.

**CONCEPTOS/TEORÍA**

[Visualizar Arquitectura De Redes Neuronales](https://adamharley.com/nn_vis/)

Se crea un nuevo modelo porque se está intentando **aislar** la primera capa del modelo preentrenado para **visualizar las activaciones** (mapas de características) de esa capa, sin necesidad de pasar por el resto de las capas del modelo original.

### **Motivos para crear un nuevo modelo:**

1. **Acceso directo a las activaciones de una capa específica**:
   * El modelo original está entrenado para realizar una tarea (como clasificación de imágenes), y contiene varias capas. Si solo deseas las activaciones de la primera capa, no es necesario procesar la imagen a través de todas las capas.
   * Al crear un nuevo modelo con una nueva capa de entrada y la primera capa del modelo preentrenado, estás obteniendo las salidas directas de esa capa (sin involucrar las demás capas).
2. **Separar el procesamiento**:
   * El nuevo modelo permite **modificar la entrada** o **ajustar la arquitectura** sin afectar el modelo preentrenado.
   * La nueva entrada se asegura de que las imágenes tengan el formato adecuado (en este caso, (225, 400, 3)), y puedes experimentar con distintas capas o activaciones de manera independiente al modelo original.
3. **Visualización de activaciones intermedias**:
   * Para visualizar los mapas de características, necesitas acceder a las **salidas intermedias** del modelo. El modelo original normalmente produce una salida final (por ejemplo, una clasificación), por lo que es necesario crear un modelo que detenga el proceso en la capa que te interesa (en este caso, la primera capa).
   * Este enfoque es común para **inspeccionar lo que ha aprendido el modelo** en las primeras capas, las cuales suelen detectar patrones simples como bordes, colores o texturas.

[Normalización En CNN](https://jorgeiblanco.medium.com/por-qu%C3%A9-la-normalizaci%C3%B3n-es-clave-e-importante-en-machine-learning-y-ciencia-de-datos-4595f15d5be0)

Normalizar los datos en una CNN con regresión es fundamental para mejorar el rendimiento y estabilidad del modelo. La normalización ajusta los valores de entrada y salida a un rango uniforme, lo que facilita el aprendizaje de los parámetros de la red y evita que los pesos se vuelvan demasiado grandes o pequeños, lo que podría causar problemas de convergencia.

### **Funciones principales de la normalización en una CNN con regresión:**

1. **Acelera el entrenamiento**: La normalización ayuda a que el modelo converja más rápido al asegurarse de que los gradientes se mantengan en un rango estable. Esto evita que el modelo tarde demasiado en aprender.
2. **Previene la saturación de las activaciones**: Cuando los datos de entrada están en un rango muy amplio, las activaciones de las neuronas en capas como ReLU o sigmoide pueden saturarse, lo que impide que los gradientes fluyan adecuadamente y ralentiza el aprendizaje.
3. **Evita el problema de escala en los pesos**: La normalización permite que todos los parámetros del modelo aprendan en la misma escala, lo que garantiza que el optimizador no se vea afectado por diferencias grandes entre las características.
4. **Mejora la precisión**: Mantener las entradas y las salidas en un rango similar permite que la red generalice mejor y realice predicciones más precisas.

### **Puntos a tener en cuenta al normalizar los datos:**

1. **Normalización de entrada**: Las imágenes de entrada deben ser escaladas a un rango uniforme (por ejemplo, entre 0 y 1 o -1 y 1). Esto se puede lograr dividiendo los valores de los píxeles por un valor constante, como 255 si los valores están en [0, 255].
2. **Normalización de salida**: En el caso de la regresión, los valores de las etiquetas (en tu caso, la cantidad de granos de maíz) deben ser escalados al mismo rango en el que se esperan las predicciones (por ejemplo, utilizando MinMaxScaler para llevarlos a un rango de [0, 1] o [0, 19], dependiendo del caso).
3. **Desnormalización al final**: Después de la predicción, es importante desnormalizar los valores de salida para interpretarlos correctamente en el dominio original. Esto significa revertir el proceso de normalización para que los valores predichos puedan compararse con las etiquetas originales.
4. **Considerar el rango de las etiquetas**: Si las etiquetas están normalizadas en un rango entre 0 y 19, el modelo debe ser capaz de producir salidas dentro de este rango. Si las predicciones salen fuera del rango esperado, es probable que el modelo necesite ajustes en la función de pérdida, las capas, o el proceso de normalización.

Normalizar correctamente tanto las entradas como las salidas es crucial para obtener predicciones precisas en una tarea de regresión con CNN.

Se intentó cambiar el código a regresión.

**Se eliminó el One-Hot Encoding:** Elimina la conversión de etiquetas y\_train y y\_test a One-Hot Encoding, ya que en la regresión la salida será un valor continuo en lugar de una categoría.

**Se modifica la capa de salida:** Cambia la capa de salida de Dense(num\_classes, activation='softmax') a Dense(1) sin función de activación (ya que la salida será un número continuo).

**Se cambia la función de pérdida:** Sustituye categorical\_crossentropy por mean\_squared\_error, que es más adecuada para tareas de regresión.