Reconocimiento de imagen por una red neuronal mediante Deep Network Designer

Ceballos Barrios Martin Camilo - Quintana Fuentes Jose Daniel
Procesamiento de Contenido Multimedia
Universidad del Magdalena
Santa Marta, Magdalena
martinceballoscb@unimagdalena.edu.co
josequintanadf@unimagdalena.edu.co

***Resumen*—En este laboratorio, se realizó el reconocimiento de imágenes, por medio de IA teniendo como protagonistas, fotos de perros, gatos, caballos y gallinas. Se realizó por medio de una red neuronal preentrenada, se entrenó para nuestra conveniencia para así tener resueltas nuestra búsqueda dentro de la serie de imágenes propuestas.**

Palabras: reconocimiento, Inteligencia Artificial, preentrenada.

***Abstract*—In this laboratory, image recognition was performed by means of AI having as protagonists, photos of dogs, cats, horses and chickens. It was performed by means of a pre-trained neural network, trained for our convenience in order to solve our search within the series of proposed images.**

Keywords: recognition, Artificial Intelligence, pre-traine.

OBJETIVOS:

- → Aprender a manejar IA.
- → Búsqueda de imágenes concretas.
- → Entrenamiento de redes neuronales.
- → Obtención de resultados según códigos propuestos.

I. MODELO TEORICO

Deep Learning o aprendizaje profundo es una forma de aprendizaje automático, en el que una máquina intenta imitar al cerebro humano utilizando redes neuronales artificiales con más de tres capas que le permiten hacer predicciones con una gran precisión.
Las redes neuronales, y más concretamente las redes neuronales artificiales (ANN – Artificial Neural Network), imitan al cerebro humano utilizando un conjunto de algoritmos. A un nivel muy básico, una neurona, de una red neuronal, consta de cuatro componentes principales: entradas, pesos, un sesgo y una salida [1].

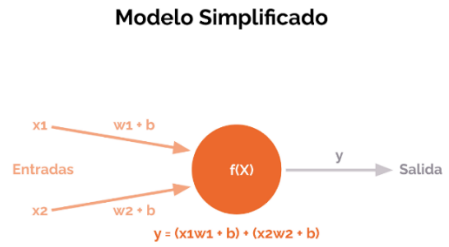


Fig 1. Modelo simplificado de una neurona artificial. Tomada de [1].

Se le llama Deep Learning o aprendizaje profundo simplemente cuando una red neuronal tiene más de tres capas.

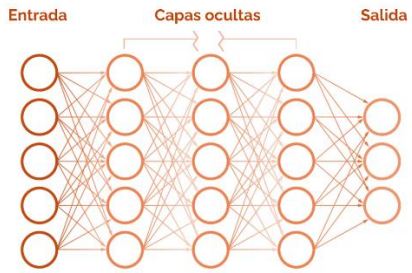


Fig 2. Red neuronal profunda. Tomada de [1].

Los tipos más simples de redes neuronales son los previamente mencionados y son redes neuronales artificiales (ANN) o redes neuronales multicapa.
Hay dos tipos de redes neuronales que suelen ser muy utilizadas que son las redes neuronales convolucionales (CNN – Convolutional neural network) y las redes neuronales recurrentes (RNN – Recurrent Neural Networks).

Redes neuronales convolucionales (CNN)

Una red neuronal convolucional (CNN o ConvNet) es una arquitectura de red para Deep Learning que aprende directamente de los datos, sin necesidad de extraer características manualmente [2].
Estas redes son especialmente útiles para encontrar patrones en imágenes para reconocer objetos, caras y escenas. Además, resultan eficaces para clasificar datos sin imágenes, tales como datos de audio, series temporales y señales.
Las aplicaciones que utilizan reconocimiento de objetos y visión artificial, tales como las aplicaciones para vehículos autónomos y para reconocimiento facial, dependen en gran medida de CNN.
El empleo de CNN con Deep Learning es popular debido a tres factores importantes. Las CNN:

- Aprenden características directamente sin necesidad de extraerlas manualmente.
- Generan resultados de reconocimiento altamente precisos.
- Se pueden volver a entrenar para nuevas tareas de reconocimiento, lo que permite aprovechar las redes preexistentes.

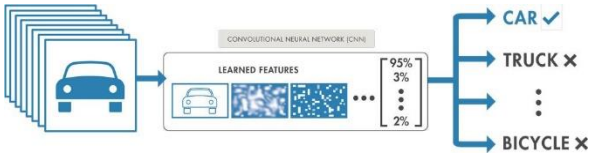


Fig 3. Flujo de trabajo de Deep Learning. Las imágenes se envían a CNN, que aprende las características y clasifica los objetos automáticamente. Tomada de [2].

Las CNN proporcionan una arquitectura óptima para descubrir y aprender características principales en imágenes y datos de series temporales [2]. Las CNN son una tecnología clave en aplicaciones tales como:

- **Imágenes médicas**: las CNN pueden explorar miles de informes patológicos para detectar visualmente la presencia o ausencia de células cancerosas en las imágenes.
- **Procesamiento de audio**: la detección de palabras clave se consigue utilizar en cualquier dispositivo con un micrófono para detectar cuándo se pronuncia una palabra o frase determinada (como, por ejemplo: "Oye Siri"). Las CNN pueden aprender y detectar con precisión la palabra clave e ignorar todas las demás frases, independientemente del entorno.
- **Detección de señales de stop**: la conducción autónoma se fundamenta en CNN para detectar con exactitud la presencia de una señal u otro objeto y tomar decisiones establecidas en el resultado.

- **Generación de datos sintéticos**: utilizando redes generativas antagónicas (GAN), se pueden producir nuevas imágenes para su uso en aplicaciones de Deep Learning, tales como reconocimiento facial y conducción autónoma.

Redes Neuronales Recurrentes (RNN)

Las redes neuronales demandantes trabajan con datos secuenciales o de series temporales. Se utilizan mucho para la traducción de idiomas, el procesamiento del lenguaje natural y el reconocimiento de voz. Es la base de muchas aplicaciones populares como Google Translate o Siri [1].
La gran diferencia de las otras redes neuronales es que tienen "memoria". Las salidas de las neuronas son utilizadas de nuevo la próxima vez que se corre el modelo [1].
Normalmente las redes neuronales profundas tradicionales asumen que las entradas y las salidas son independientes entre sí.

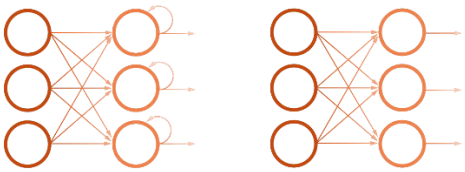


Fig 4. Retropropagación. Tomado de [1].

A parte de la retropropagación normal igualmente tenemos la retropropagación en el tiempo para este tipo de red neuronal. Todos esos pasos de tiempo se suman rápidamente que crean capas adicionales creando una red neuronal muy profunda en muy poco tiempo.

Arquitecturas RNN especializadas
Cuando estamos construyendo RNNs, el estado de las etapas anteriores se diluye con el tiempo. Esto puede ser un problema, por ejemplo, cuándo es aprendizaje de estructuras de frases y por ejemplo las iniciales palabras de la frase son muy importantes []. Esto se puede contrarrestar con células de memoria. Existe dos tipos de célula que puedes usar:

- Célula LSTM: Long short-term memory. Conserva estados separados a corto y largo plazo.
- Célula GRU – Gated Recurrent Unit. Célula LSTM resumida que funciona casi igual de bien.

Deep Learning Toolbox

Deep Learning Toolbox™ proporciona las herramientas para diseñar e implementar redes neuronales profundas con algoritmos, modelos previamente entrenados y apps. Puede utilizar redes neuronales convolucionales (ConvNet y CNN) y redes de memoria de corto-largo plazo (LSTM) para realizar tareas de clasificación y regresión en imágenes, series temporales y datos de texto. Con la app Deep Network Designer, puede diseñar, analizar y entrenar redes gráficamente [3].

Deep Network Designer

Deep Network Designer para adecuar una red preentrenada para clasificar un nuevo conjunto de imágenes. Este proceso se denomina transferencia del aprendizaje y suele ser mucho más rápido y fácil que entrenar una nueva red, ya que admite aplicar las características aprendidas a una nueva tarea con menos imágenes de entrenamiento. Para preparar una red para la transferencia del aprendizaje de forma interactiva, utilice Deep Network Designer [4].

Squeezenet

SqueezeNet es una red neuronal convolucional de 18 capas de profundidad. Alcanza cargar una versión preentrenada de la red entrenada con más de un millón de imágenes de la base de datos ImageNet [5]. La red preentrenada puede clasificar las imágenes en 1000 categorías de objetos, como teclado, ratón, lápiz y muchos animales. Como resultado, la red ha aprendido representaciones ricas en características para una amplia gama de imágenes. Esta función devuelve una red SqueezeNet v1.1, que tiene una precisión similar a la de SqueezeNet v1.0 pero que requiere menos operaciones de coma flotante por

predicción [6]. La red tiene un tamaño de entrada de imagen de 227 por 227.

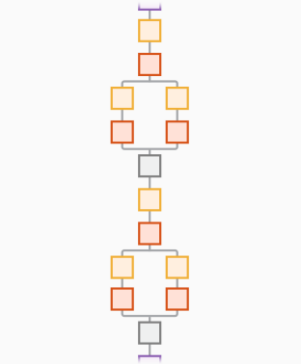
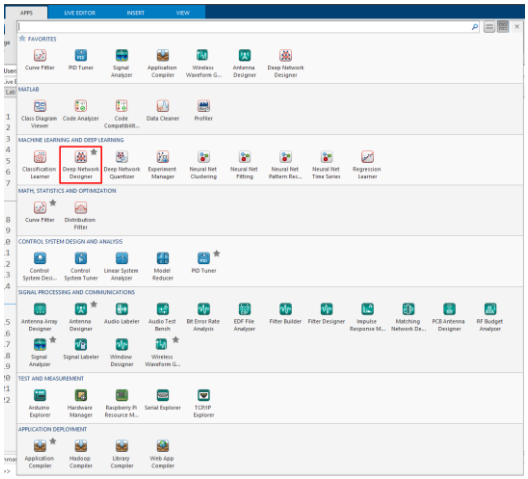


Fig 5. SqueezeNet convolutional neural network. Tomado de [7].

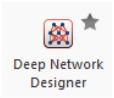
II. DESARROLLO

Pasos del desarrollo de entrenamiento de la red neuronal utilizando Deep Nwetwork Designer. Para este entrenamiento se seleccionarán unos cuantos animales (caballo, gallina, gato, perro) para realizar el entrenamiento

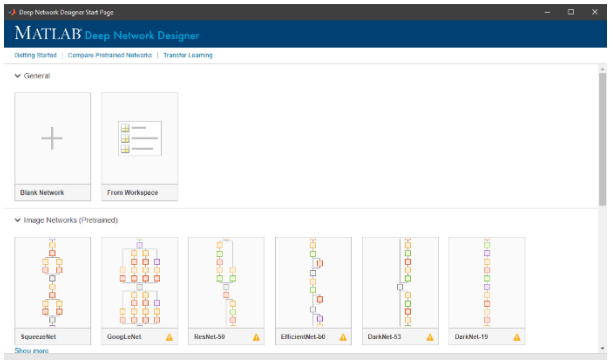
La app utilizada para este laboratorio se encuentra en el apartado de “APPS” en Matlab.



Una vez ubicada la aplicación la abrimos.

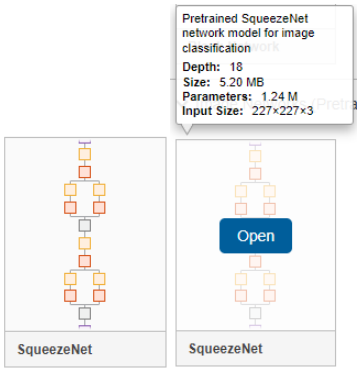


Se despliega una ventana donde podemos crear nuestra red neuronal desde cero o escoger una red neuronal preentreanda.

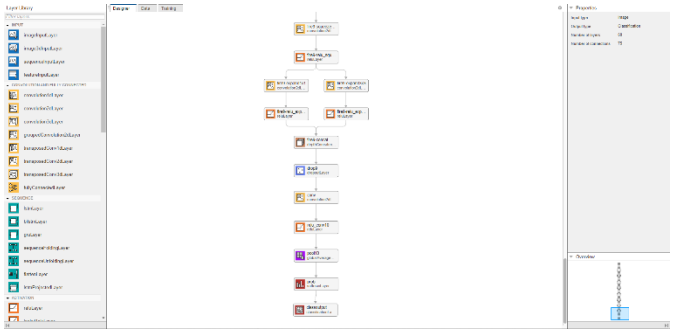


Para este laboratorio escogeremos la red neuronal más sencilla, pero de igual manera muy útil para el desarrollo de la actividad.

Seleccionando la red neuronal: SqueezeNet



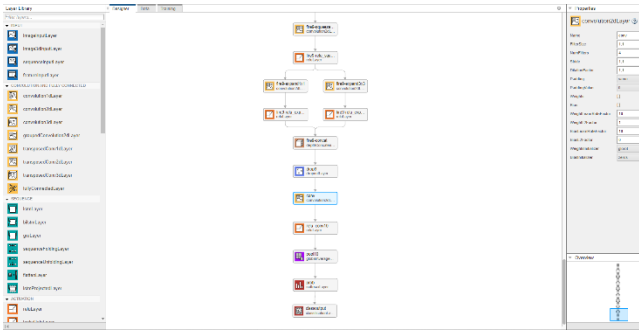
Deep Network Designer muestra una vista ampliada de toda la red en el panel Diseñador.



Las capas convolucionales de la red extraen las características de la imagen que la última capa de aprendizaje y la capa de clasificación final utilizan para clasificar la imagen de entrada. Estas dos capas, 'convolution2dLayer' y 'classificationLayer' en SqueezeNet, contienen información sobre cómo combinar las características que extrae la red en probabilidades de clase, un valor de pérdida y etiquetas predichas. Para volver a entrenar una red previamente entrenada para clasificar nuevas imágenes, se reemplaza estas dos capas con nuevas capas adaptadas al nuevo conjunto de

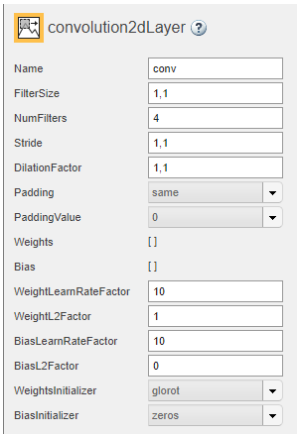
datos.

Primero modificar el bloque, convolution2dLayer.

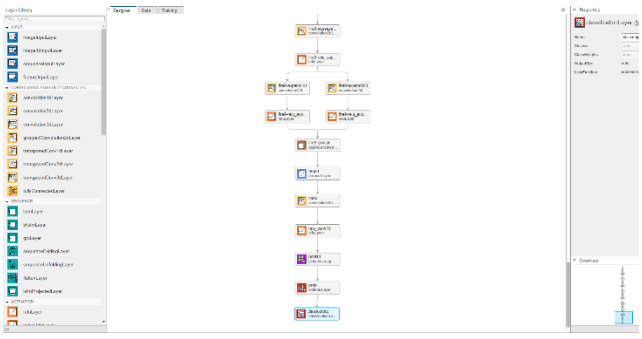


Agregamos el nuevo bloque, el numero filtros (NumFilters) seria de 4 debido a que nuestra recopilación de imágenes tiene 4 categorías de imágenes.

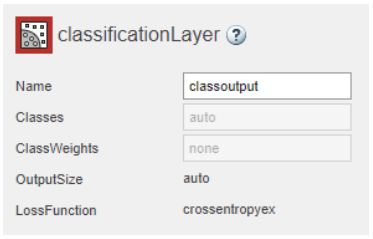
WeigthLearnRateFactor: 10
BiasLearnRateFactor: 10



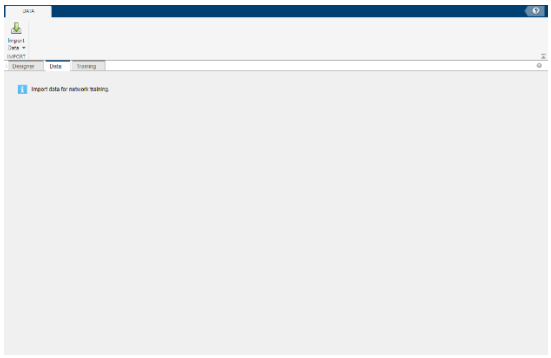
Segundo modificar el bloque, classificationLayer.



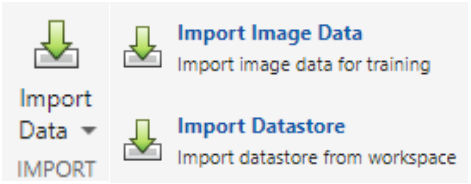
Agregamos el nuevo bloque, el número de clases (Classes) se configura automáticamente.



Importando la Data.



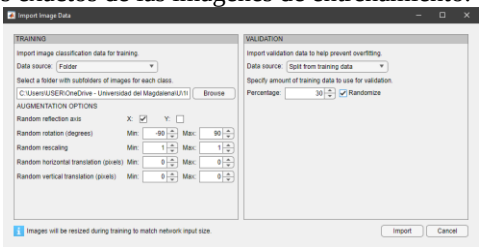
Para cargar los datos en Deep Network Designer, en la pestaña Datos, se hace clic en Importar datos > Importar datos de imagen. Se abre el cuadro de diálogo Importar datos de imagen.



En la lista Fuente de datos, se selecciona Carpeta. Se hace clic en Examinar y se selecciona la carpeta MerchData extraída.

Deep Network Designer divide los datos en 70 % de datos de entrenamiento y 30 % de datos de validación.

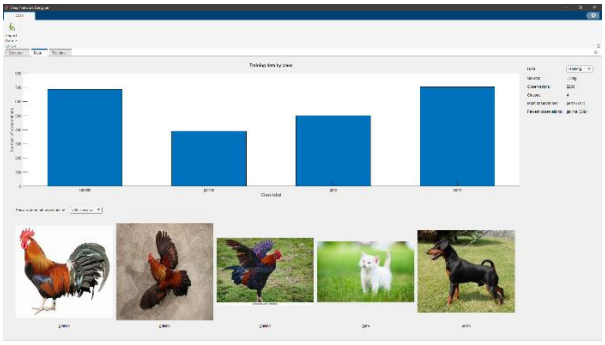
Para este ejercicio, se aplica un reflejo aleatorio en el eje x, una rotación aleatoria desde el rango [-90,90] grados. El aumento de datos ayuda a evitar que la red se sobreajuste y memorice los detalles exactos de las imágenes de entrenamiento.

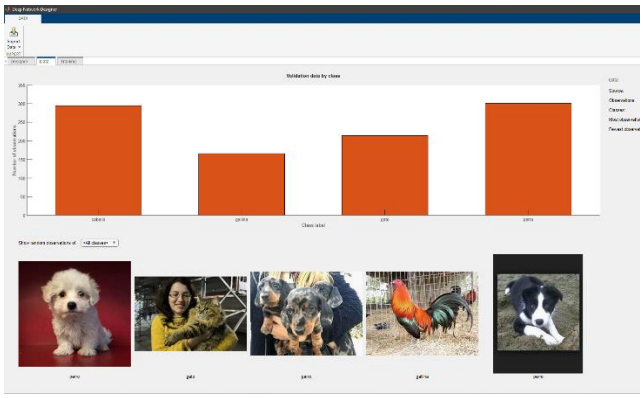


Visualizar datos

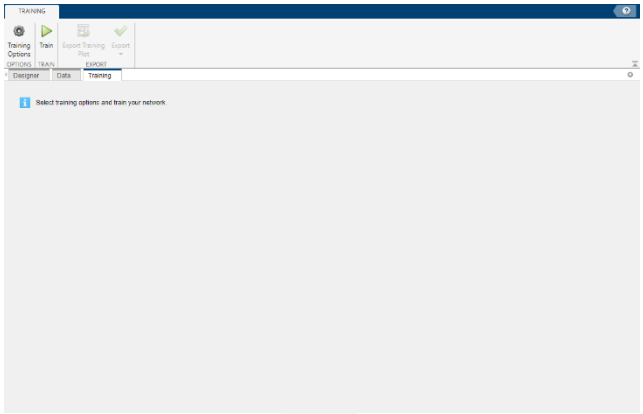
Con Deep Network Designer, se puede inspeccionar visualmente

la distribución de los datos de entrenamiento y validación en la pestaña Datos. También se puede ver observaciones aleatorias y sus etiquetas como una simple verificación antes del entrenamiento. En este ejercicio, hay cuatro clases en el conjunto de datos.





Pestaña Entrenamiento (Tranning) y se hace clic en Opciones de entrenamiento (Tranning Options).



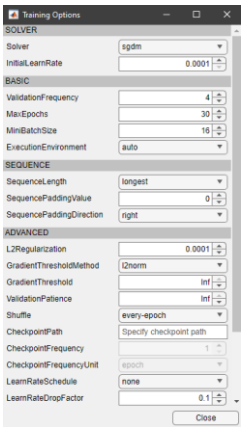
- Es necesario establecer la tasa de aprendizaje inicial (InitialLearnRate) en un valor pequeño para ralentizar el aprendizaje en las capas transferidas.

- Luego la frecuencia de validación para que la precisión de los datos de validación se calcule una vez cada época.

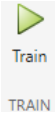
- Ajustar un pequeño número de épocas (MaxEpochs). Una época es un ciclo de entrenamiento completo en todo el conjunto de datos de entrenamiento. Para el aprendizaje por transferencia, no es necesario entrenar durante tantas épocas.

- Se especifica el tamaño del mini-lote (MinbatchSize), es decir, cuántas imágenes usar en cada iteración. Para garantizar que se utilice todo el conjunto de datos durante cada época (MaxEpochs), establezca el tamaño del mini-lote (MinbatchSize), para dividir uniformemente el número de muestras de entrenamiento.

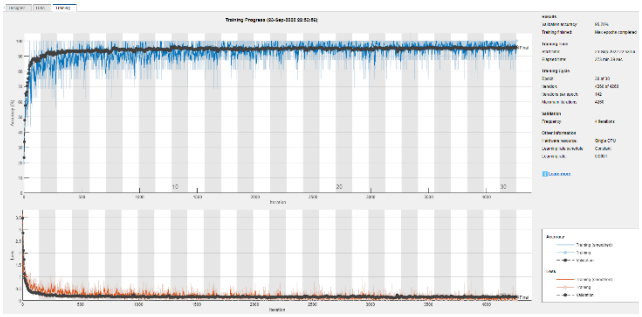
Para este ejercicio, se configura InitialLearnRate en 0.0001, ValidationFrequency en 4 y MaxEpochs en 30. Como hay 55 observaciones, configure MiniBatchSize en 16.



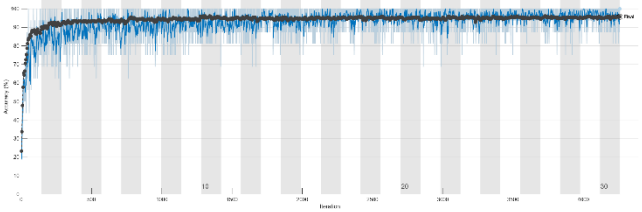
Para entrenar la red con las opciones de entrenamiento especificadas, se hace clic en Cerrar y luego se hace clic en Entrenar (Train).



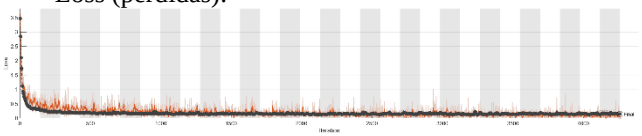
Deep Network Designer permite visualizar y monitorear el progreso del entrenamiento. Luego se puede editar las opciones de entrenamiento y volver a entrenar la red, si es necesario.



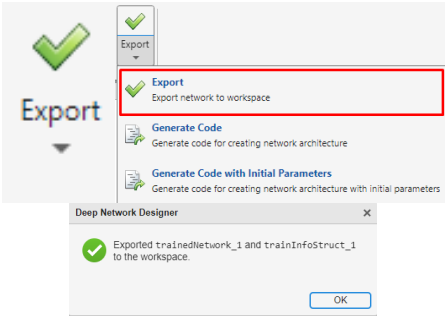
Accuracy (presición):



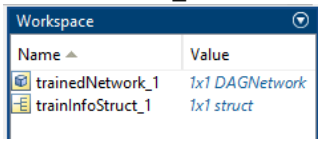
Loss (pérdidas):



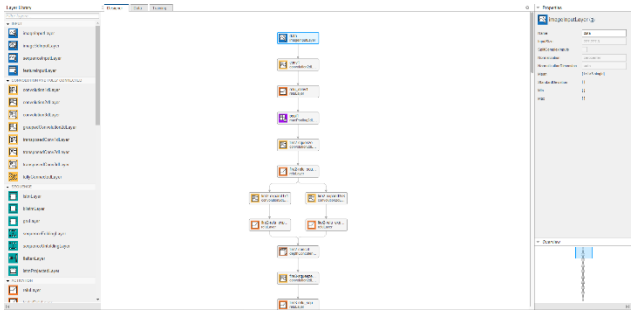
Para exportar la arquitectura de red con los pesos entrenados (trained weights), en la pestaña Entrenamiento, se selecciona Exportar > Exportar red y resultados entrenados.



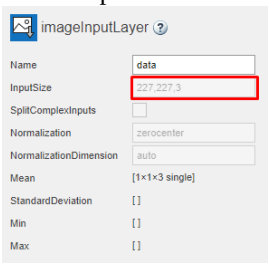
Deep Network Designer exporta la red entrenada como la variabletrainedNetwork_1 y la información de entrenamiento como la variable trainInfoStruct_1.



Deep Network Designer cambia el tamaño de las imágenes durante el entrenamiento para que coincidan con el tamaño de entrada de la red.



Para ver el tamaño de entrada de la red, en el panel Diseñador, se selecciona imageInputLayer (primera capa). Esta red tiene un tamaño de entrada de 227 por 227.



Comprobando la validez de la red entrenada.

Prueba de red con imagenes seleccionadas de la misma recopilación de imágenes que le red utilizo para entrenar.

```
I = imread("C:\Users\USER\OneDrive - Universidad del Magdalena\U\10mo Semestre\Procesamiento Multimedia\Lab\Learning\img\perro\OIP-__YuIXH3iAC100zGQFpC-AHaE8.jpeg");
I = imresize(I, [227 227]);
imshow(I)

[YPred,probs] = classify(trainedNetwork_1,I);
label = YPred;
title(string(label)+", "+num2str(100*max(probs),3)+ "%")
```

Resultado de la clasificación de la imagen puesta a prueba por la red neuronal entrenada.


```
I2 = imread("C:\Users\USER\OneDrive - Universidad del Magdalena\U\10mo Semestre\Procesamiento Multimedia\Lab\Learning\img\gato\170.jpeg");
I2 = imresize(I2, [227 227]);
imshow(I2)

[YPred,probs] = classify(trainedNetwork_1,I2);
label2 = YPred;
title(string(label2)+", "+num2str(100*max(probs),3)+ "%")
```

Resultado de la clasificación de la imagen puesta a prueba por la red neuronal entrenada.

Prueba de red con imagenes aleatorias descargas de internet.

```
I3 = imread("C:\Users\USER\OneDrive - Universidad del Magdalena\U\10mo Semestre\Procesamiento Multimedia\Lab\Learning\gallina2.jpg");
I3 = imresize(I3, [227 227]);
imshow(I3)

[YPred,probs] = classify(trainedNetwork_1,I3);
label3 = YPred;
title(string(label3)+", "+num2str(100*max(probs),3)+ "%")
```


```
I4 = imread("C:\Users\USER\OneDrive - Universidad del Magdalena\U\10mo Semestre\Procesamiento Multimedia\Lab\Learning\c
I4 = imresize(I4, [227 227]);
imshow(I4)

[YPred,probs] = classify(trainedNetwork_1,I4);
label4 = YPred;
title(string(label4)+", "+num2str(100*max(probs),3)+ "%")
```

III. CONCLUSIONES

Finalmente, se logró realizar el reconocimiento de imagen, para el caso de este informe, como antes se mencionó, las categorias de las imágenes, son: caballo, gallina, gato y perro. Ademas, teniendo una red neuronal preentrenada mediante Deep Network Designer se es posible la modificación de esta para poder desarrollar nuestra propia red neuronal y para así realizar el entrenamiento con la data deseada. Asi pues, luego de un largo entrenamiento de la red se consiguio que reconociera las categorias deseadas las cuales se les enseña anteriormente.

Referencias

[1] Datademia, «¿Qué es Deep Learning y qué es una red neuronal?,» [En línea]. Available: https://datademia.es/blog/que-es-deep-learning-y-que-es-una-red-neuronal.

[2] Mathworks, «Redes neuronales convolucionales,» [En línea]. Available: https://la.mathworks.com/discovery/convolutional-neural-network-matlab.html|.

[3] MathWorks, «Deep Learning Toolbox,» [En línea]. Available: https://la.mathworks.com/products/deep-learning.html.

[4] MathWorks, «Introducción a Deep Network Designer,» [En línea]. Available: https://la.mathworks.com/help/deeplearning/gs/get-started-with-deep-network-designer.html.

[5] «ImageNet,» [En línea]. Available: http://www.image-net.org.

[6] F. N. Iandola, «SqueezeNet,» [En línea]. Available: https://github.com/forresti/SqueezeNet.

[7] MathWorks, «squeezenet,» [En línea]. Available: https://la.mathworks.com/help/deeplearning/ref/squeezenet.html#mw_cc799cb1-b3a7-489d-978a-3dd541bb74ca_sep_mw_6dc28e13-2f10-44a4-9632-9b8d43b376fe.