Active Class #4...

Maestría en Inteligencia Artificial Aplicada

Introducción a DL y CNN

Dr. David Antonio-Torres

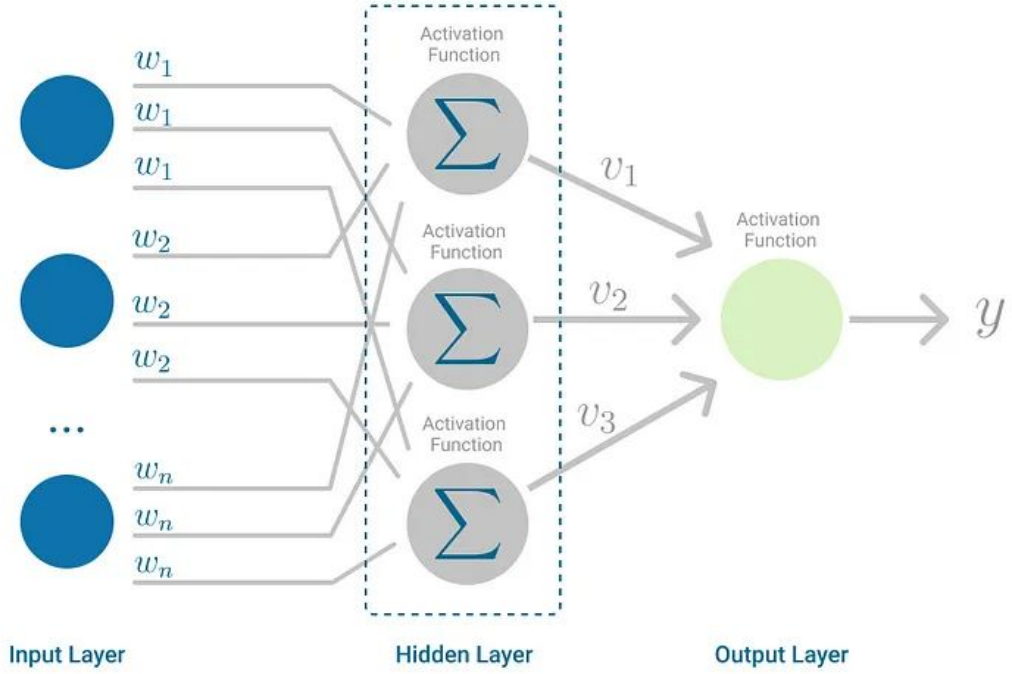
Objetivo

Revisar los conceptos de Deep Learning y Convolutional Neural Networks

Active class

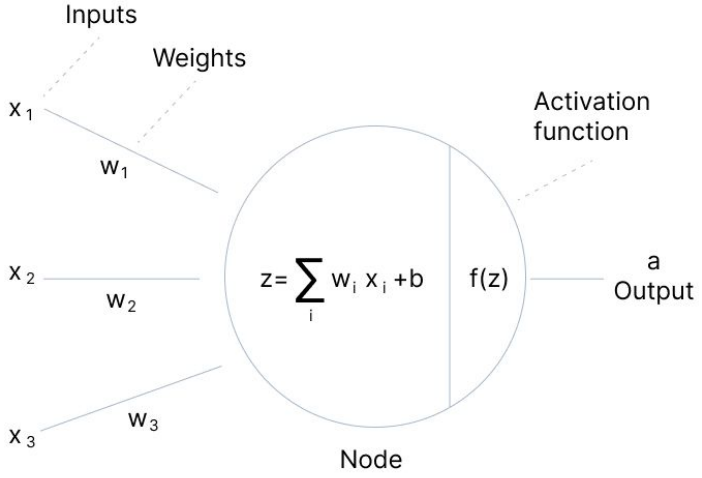
Perceptrón Multicapa

Perceptrón Multicapa



https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

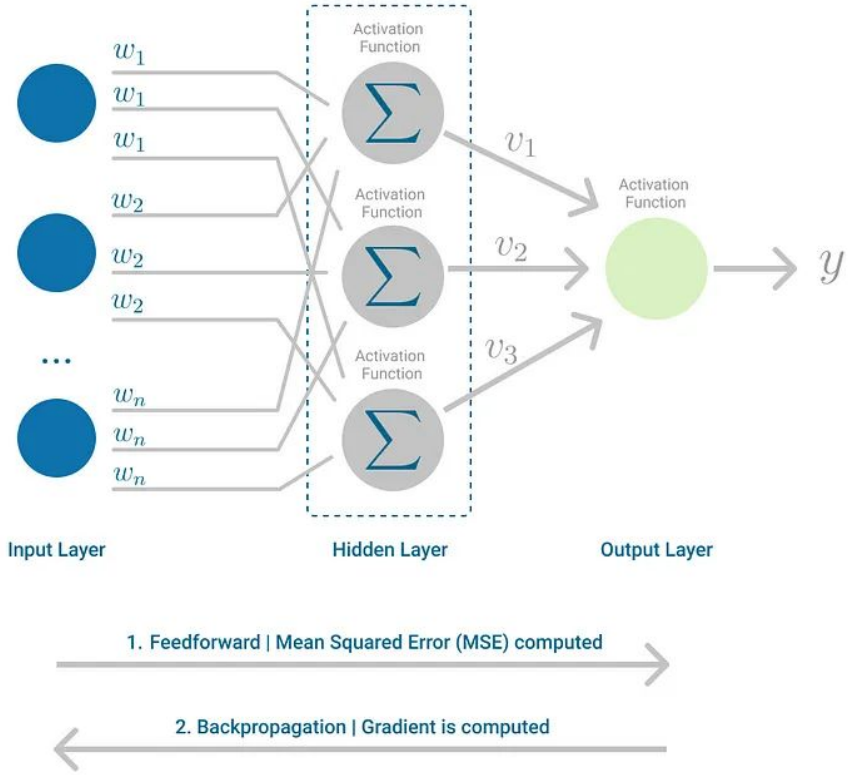
Se muestran los elementos del perceptrón y perceptrón multicapa.



V7 Labs

https://www.v7labs.com/blog/neural-networks-activation-functions

Perceptrón Multicapa

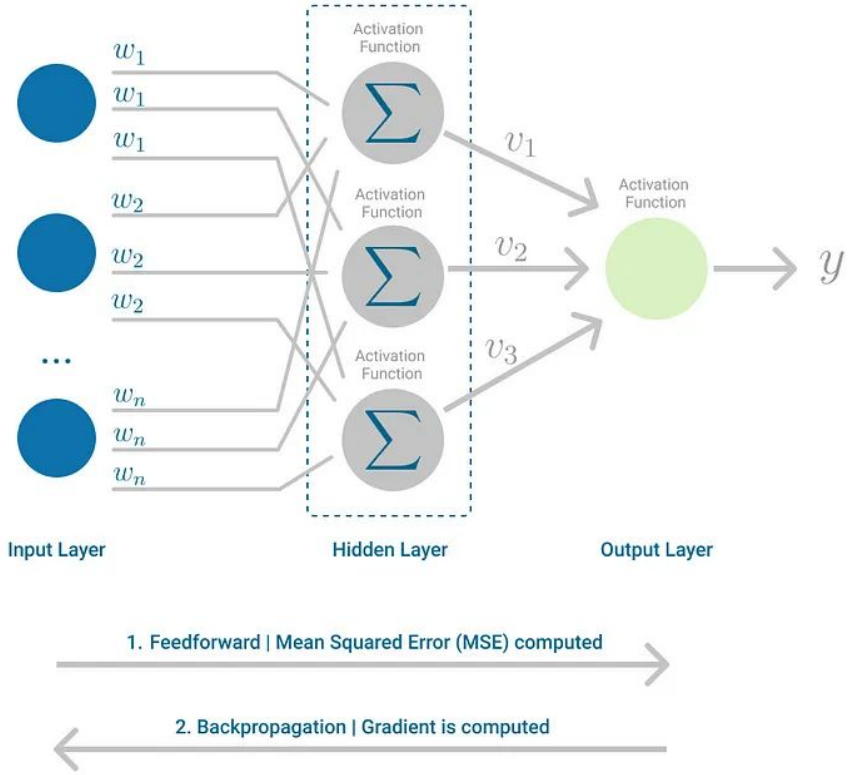


https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

El proceso de ***Feedforward*** permite que la información se propague desde las entradas hasta la salida.

La función de pérdida, ***loss function***, permite determinar qué tan cerca quedó la predicción con respecto a un resultado correcto.

Perceptrón Multicapa



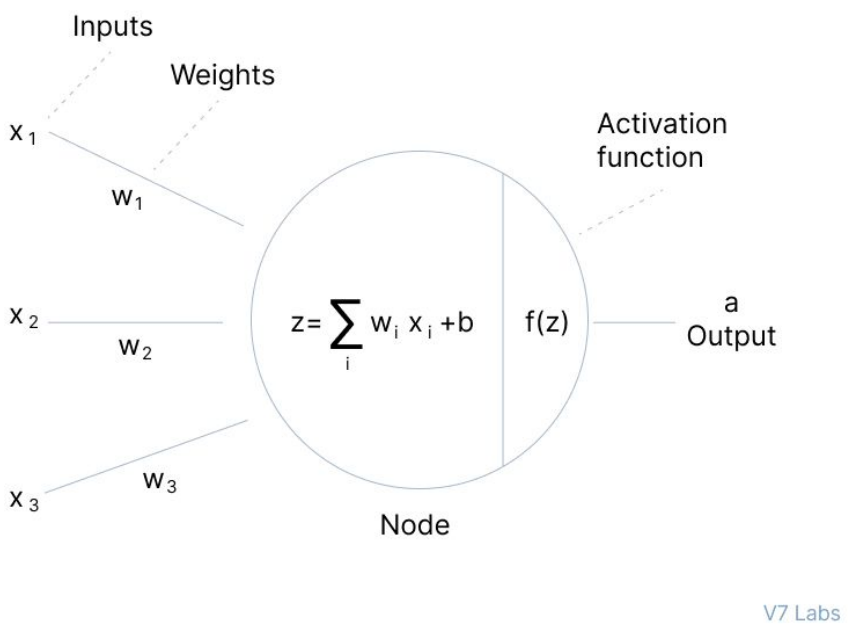
https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

El gradiente de la función de pérdida es utilizado para actualizar los parámetros de la red neuronal, partiendo desde las salidas hasta la entrada (***Backpropagation***).

Active class

Activation Functions

Funciones de Activación

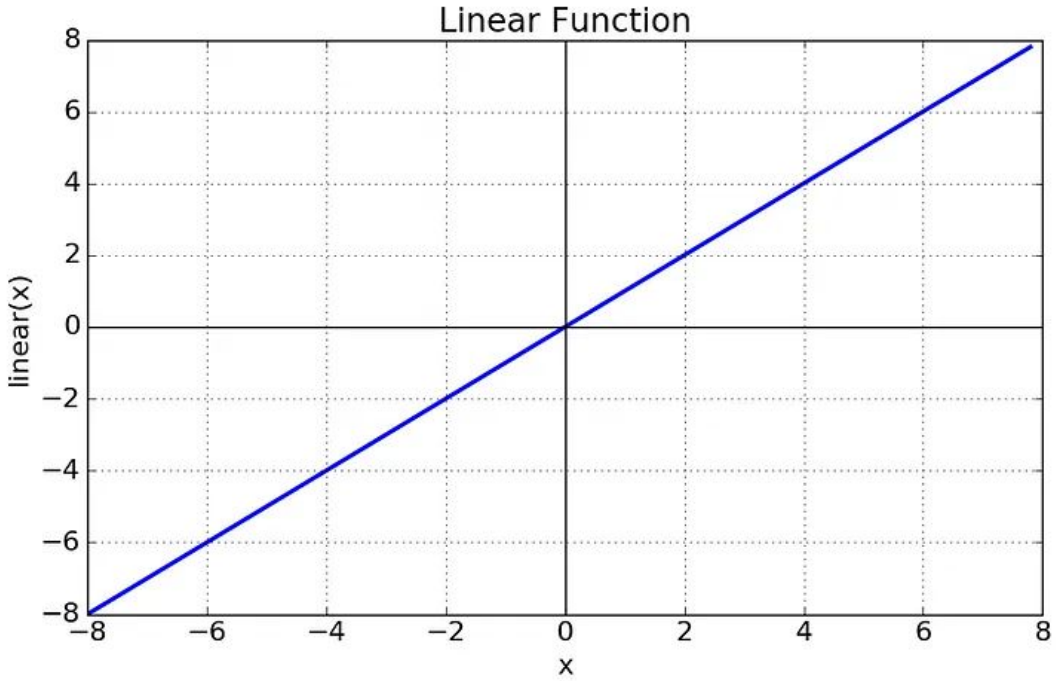


https://www.v7labs.com/blog/neural-networks-activation-functions

La función de activación tiene el objetivo de imitar que una neurona dispare o no dispare.
De la misma manera, la función de activación puede ser usada como salida de una red neuronal y definir un sí o un no.
Las funciones de activación se clasifican en dos tipos:

- Lineales
- No lineales

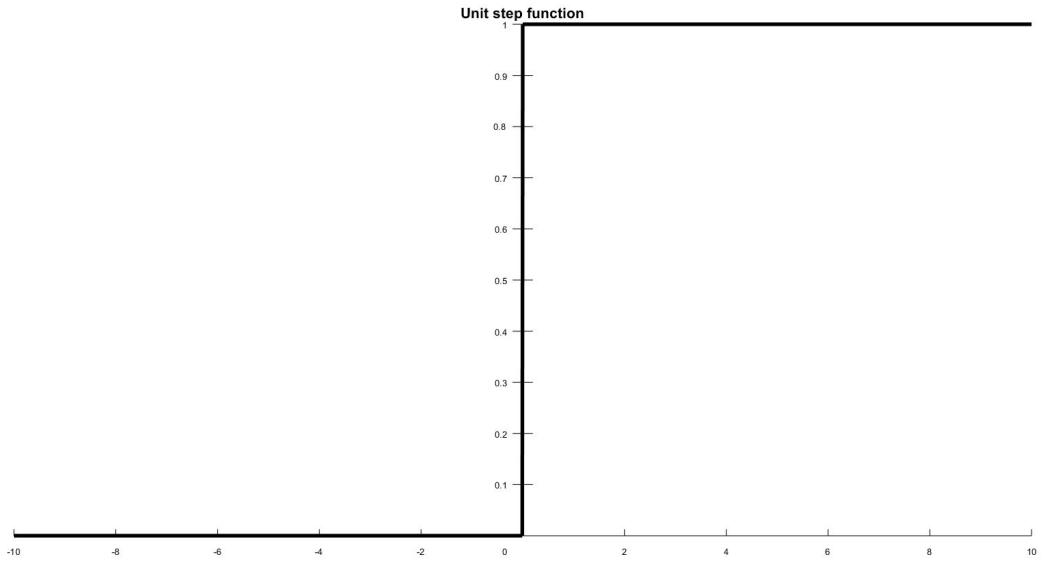
Linear Function



Dicha función se caracteriza por f(x) = x
Se utiliza como salida de la última capa de una red neuronal en aplicaciones de modelos de regresión.

https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Step Function

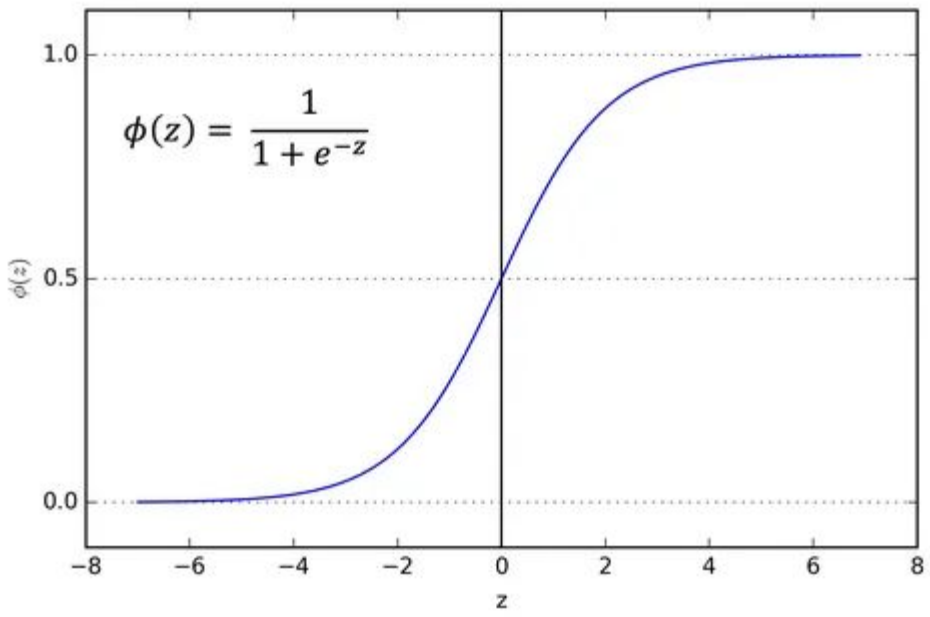


Dadas sus dos únicas salidas, se puede utilizar para fines de clasificación binaria.

$$f(x) = o \; when \; x < o,$$
$$1 \; when \; x >= o$$

https://www.oreilly.com/library/view/neural-networks-with/9781788397872/c5736f7e-91cb-4811-bed8-d746140d3eec.xhtml

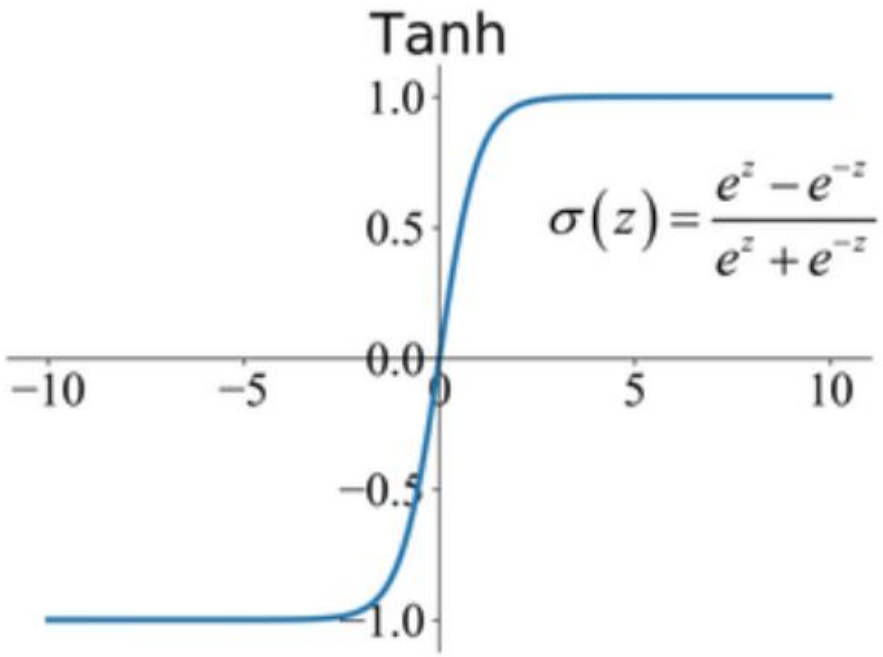
Sigmoid Function



Cuando se desea que la salida de una red neuronal esté expresada como una probabilidad, se hace uso de la función Sigmoide.

https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

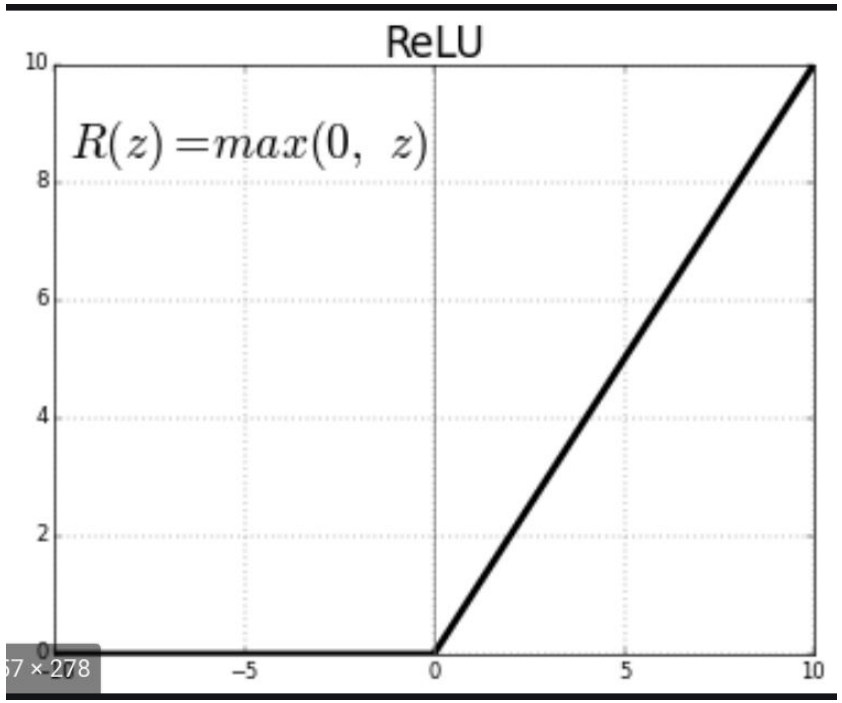
Tanh Function



La función tanh expande la salida de la función Sigmoide desde -1 hasta 1, mejorando su desempeño.
Un problema no resuelto por Tanh es “vanishing gradient” en el método de Backpropagation.

https://paperswithcode.com/method/tanh-activation

ReLu Function



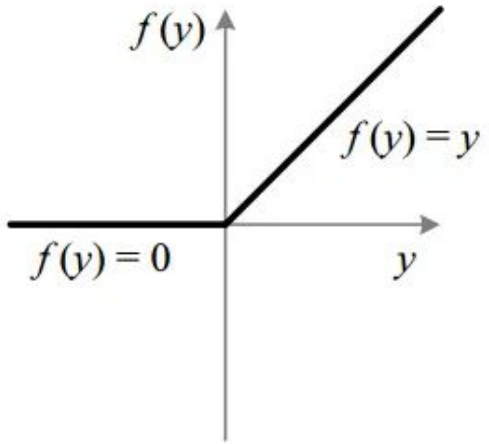
https://paperswithcode.com/method/tanh-activation

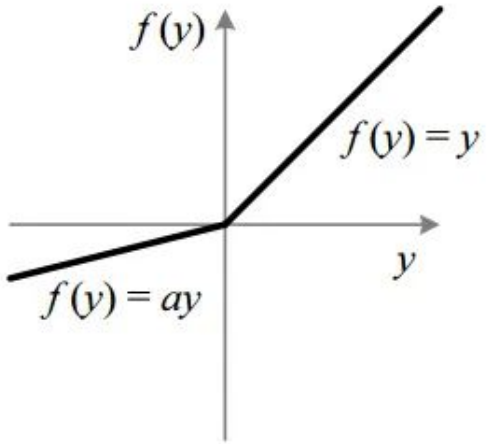
La función “Rectified Linear” es mucho más simple de implementar dada su porción lineal para valores positivos.

Es actualmente una de las funciones de activación más usadas en DL y CNN.

Se ha detectado que, debido a que, para valores negativos la salida es cero, afectando el entrenamiento de la red neuronal.

Leaky ReLu

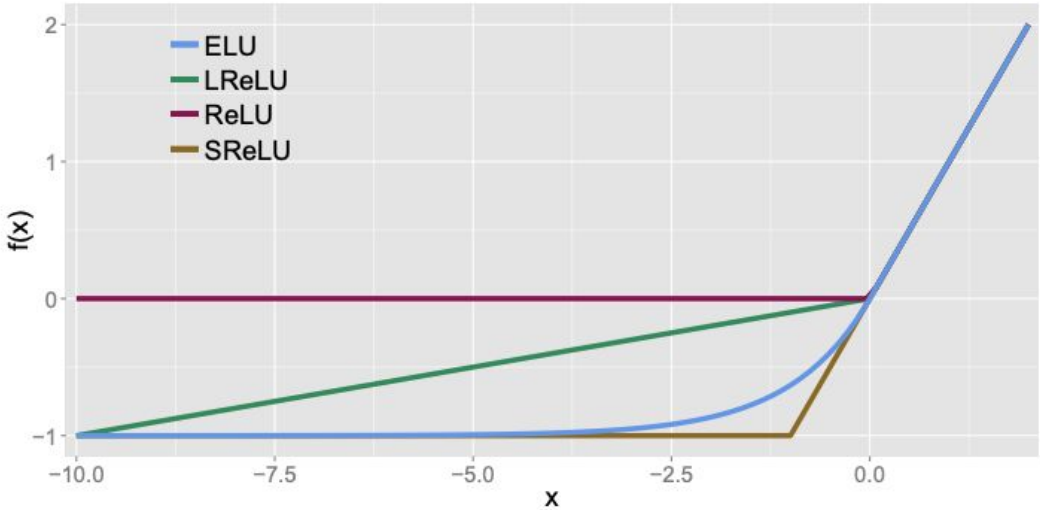




Una forma de solucionar el problema de “vanishing gradient” es evitar que la función ReLu se haga cero para entradas negativas. Así nace Leaky ReLu.

https://paperswithcode.com/method/tanh-activation

ELU



La función ELU (Exponential Linear Unit) permite mejorar la tasa de aprendizaje.

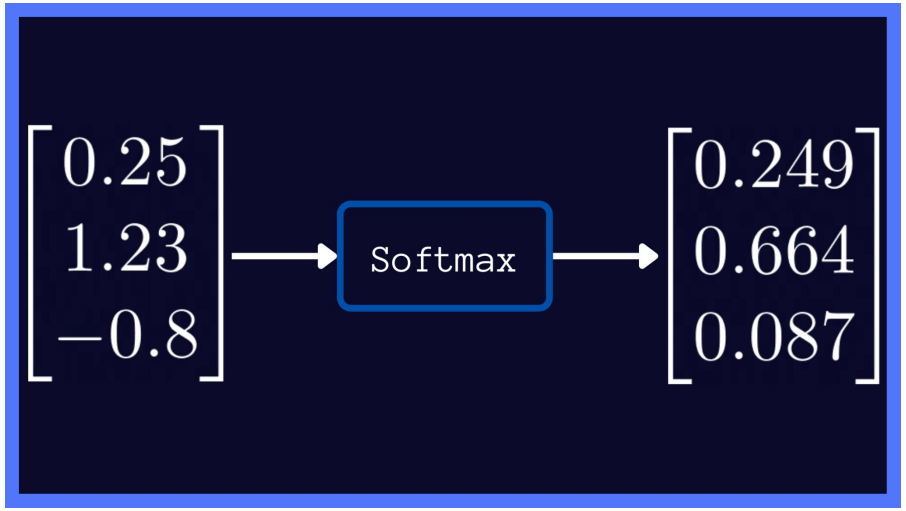
$$f(x) = x \text{ if } x > 0$$

$$\alpha(\exp(x) - 1) \text{ if } x \leq 0$$

https://paperswithcode.com/method/elu

Softmax Function

$$softmax(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$



La función Softmax toma como entrada valores no calibrados y no normalizados de la red neuronal y regresa una distribución normalizada de probabilidades para clasificación múltiple.

La distribución regresada por Softmax representa scores de confianza, los cuales suman 1. La clase predicha está asociada a la neurona que regresa el score de confianza más grande.

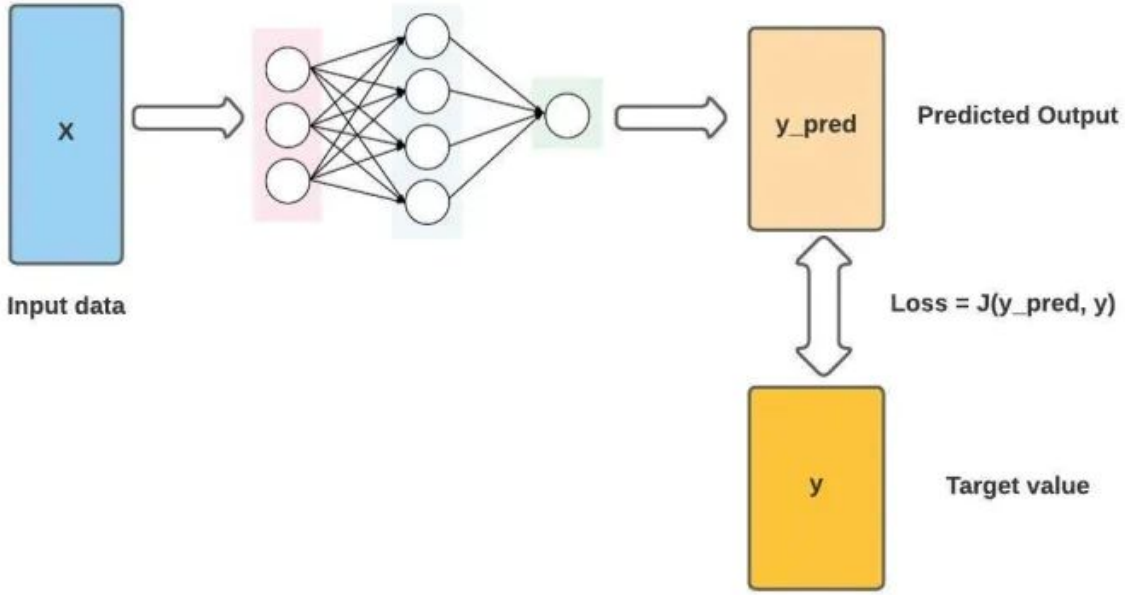
En la ecuación, z representa el vector de salidas sin procesar que regresa la red neuronal.

https://www.pinecone.io/learn/softmax-activation/

Active class

Loss Functions

Funciones de Pérdida



La función de pérdida permite determinar de una manera matemática qué tan bien el algoritmo está modelando el dataset.
En función de la aplicación se elige la función de pérdida.
Para regresión están ***Mean Squared Error*** (MSE) y ***Mean Absolute Error*** (MAE).
Para clasificación están ***Binary Cross-Entropy*** y ***Categorical Cross-Entropy***.
y_pred es representado como y-hat

https://neptune.ai/blog/pytorch-loss-functions

Funciones de Pérdida

$$MSE = \frac{1}{N} \sum_{i}^{N} (Yi - \hat{Y}i)^2$$

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |Yi - \hat{Y}i|$$

Se muestran las ecuaciones para calcular MSE y MAE.

https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/

Funciones de Pérdida

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$$

Binary Cross-Entropy se utiliza para problemas de clasificación binaria.

Compara cada una de las probabilidades predichas con la salida de la clase verdadera, la cual puede ser 0 ó 1.

Penaliza las probabilidades basándose en la distancia del valor esperado.

https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/

Funciones de Pérdida

Categorical Cross-Entropy se utiliza para clasificación multiclase.

$$\text{Loss} = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$

where k is number of classes in the data

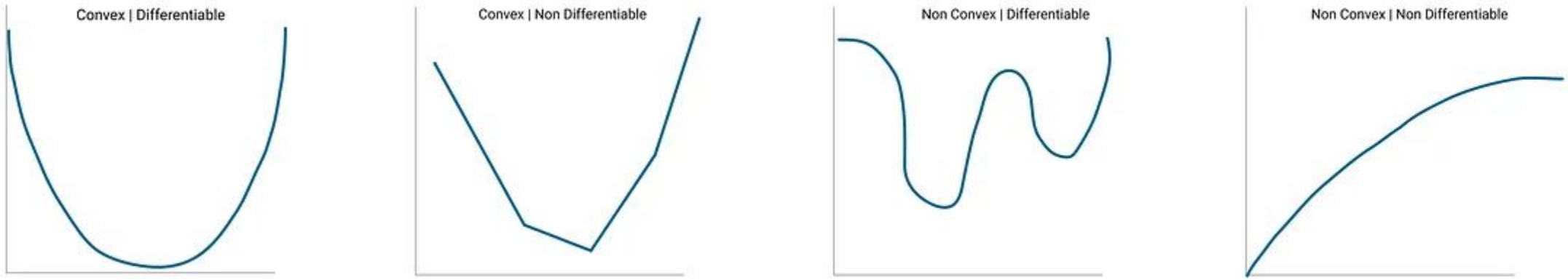
https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/

Active class

Gradient Descent

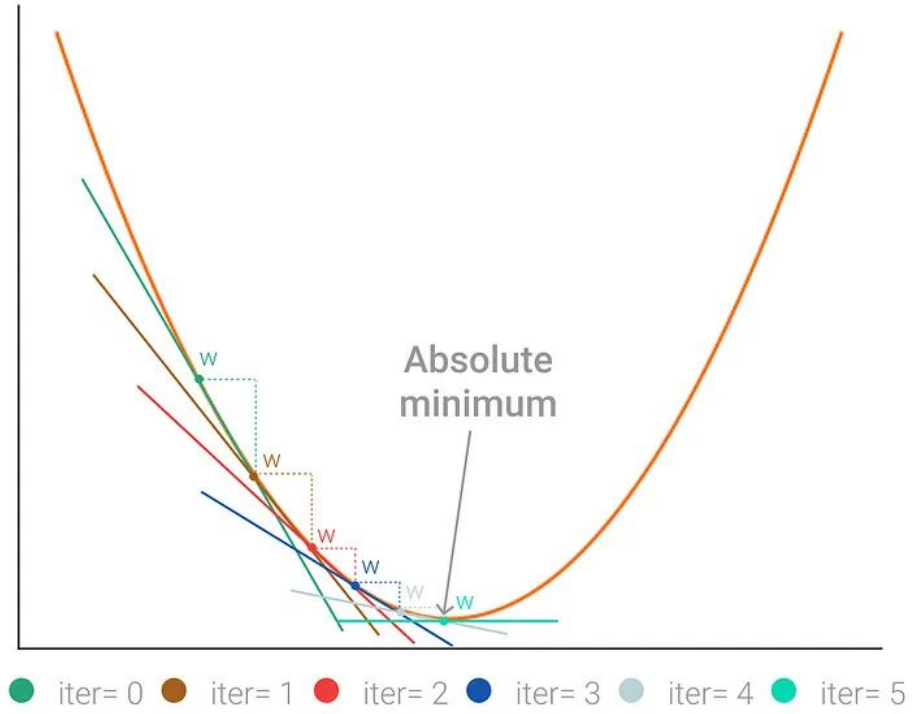
Gradient Descent

Es un método de optimización que busca reducir al mínimo la función de costo (cost function).
La función debe ser convexa y diferenciable.



https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32

Gradient Descent

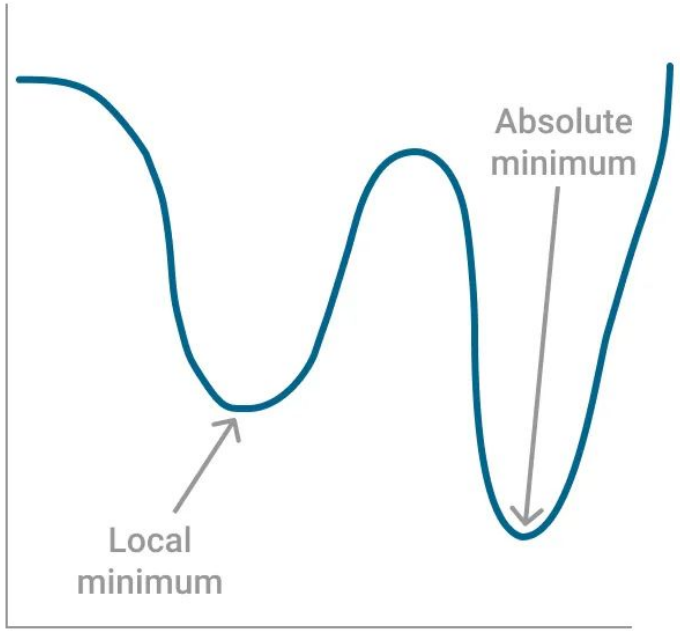


https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32

El gradiente de una función es la colección de todas sus derivadas parciales respecto a sus variables independientes.

1. Elige un punto de inicio en la función
2. Cuando el gradiente no llegue a su mínimo
   a. Calcula el gradiente negativo en ese punto
   b. Actualiza el punto inicial con el valor obtenido en el paso anterior y regresa al punto 1
3. El algoritmo ha convergido al mínimo definido

Learning Rate



El learning rate es el paso en el algoritmo de Gradient Descent que le permite llegar al mínimo global.
Si es muy grande, quizá no se llegue a encontrar el mínimo global.
Si es muy pequeño, quizá sí se encuentre el mínimo global, pero le tomará mucho tiempo al algoritmo encontrarlo.
Una limitante de Gradient Descent es que trabaja con todo el dataset.

https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32

Active class

Métodos de Regularización

Métodos de Regularización

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

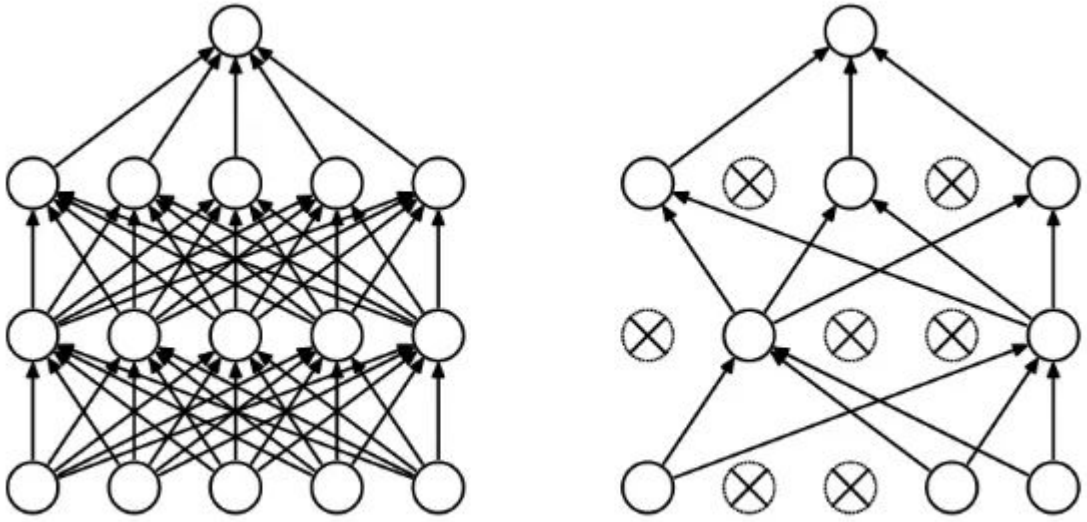
$$\text{Cost} = \underbrace{\sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Estos métodos buscan reducir el error de generalización aplicando una penalización a la función de pérdida.

La penalización se basa en los valores de los pesos y los biases de las neuronas. Pesos grandes podrían indicar que la neurona está intentando memorizar en lugar de generalizar.

https://medium.com/analytics-vidhya/l1-vs-l2-regularization-which-is-better-d01068e6658c

Métodos de Regularización



(a) Standard Neural Net (b) After applying dropout.

Dropout is otro método de regularización que permite mejorar la generalización de una red neuronal, y previniendo el overfitting, apagando un cierto porcentaje de nodos (20% a 50%) durante el entrenamiento.

https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9

Active class

Tensorflow Playground

Active class

Keras para DL

Modelo de DL

Keras nos permite crear redes neuronales de una forma sencilla agregando capas de redes neuronales e indicando cuántos nodos tiene cada capa.

Se define el modelo

```
[ ]  1 model = Sequential()
     2 model.add(Dense(units = 30, input_dim=num_pixels, activation='relu'))
     3 #model.add(Dropout(0.2))
     4 model.add(Dense(units = 10, activation='relu'))
     5 #model.add(Dropout(0.2))
     6 model.add(Dense(units = 10, activation='softmax'))
     7 model.compile(Adam(lr=0.01), loss='categorical_crossentropy', metrics=['accuracy'])
     8 print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 30)                23550

 dense_1 (Dense)             (None, 10)                310

 dense_2 (Dense)             (None, 10)                110

=================================================================
Total params: 23,970
Trainable params: 23,970
Non-trainable params: 0
_________________________________________________________________
```

Modelos Pre-entrenados

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4

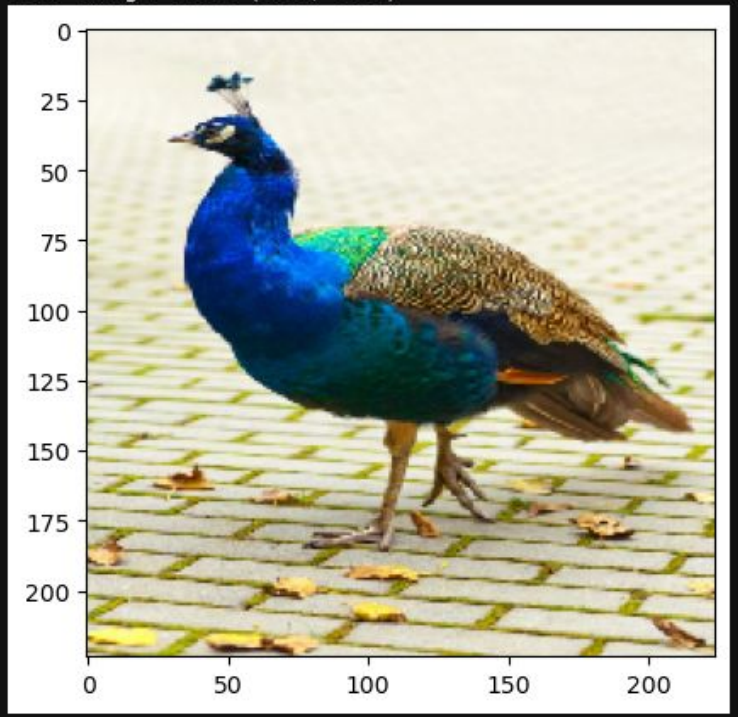
https://keras.io/api/applications/

Keras también ofrece un grupo de redes neuronales ya entrenadas que pueden ser importadas para usarlas en aplicaciones específicas.
Estos modelos han sido validados con el dataset ImageNet y algunas de ellas han sido entrenadas con este mismo dataset.

https://www.image-net.org/update-mar-11-2021.php

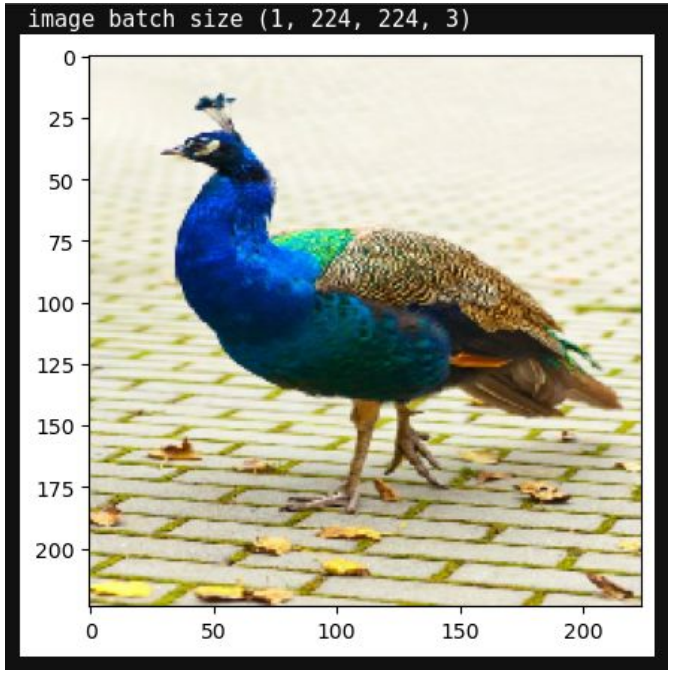
Modelos Pre-entrenados

```
[15]: filename = 'peacock.jpg'
      ## load an image in PIL format
      original = load_img(filename, target_size = (224, 224))
      print('PIL image size',original.size)
      plt.imshow(original)
      plt.show()

      PIL image size (224, 224)
```



Se va a hacer uso de la red entrenada ResNet50 y se le va a pedir al modelo que identifique al objeto en la imagen.

Modelos Pre-entrenados




```python
#processed_image = ResNet50.preprocess_input(image_batch.copy())
processed_image = tf.keras.applications.resnet50.preprocess_input(image_batch.copy())

# create resnet model
resnet_model = ResNet50(weights = 'imagenet')

# get the predicted probabilities for each class
predictions = resnet_model.predict(processed_image)

# convert the probabilities to class labels
label = decode_predictions(predictions)

print(label)
```

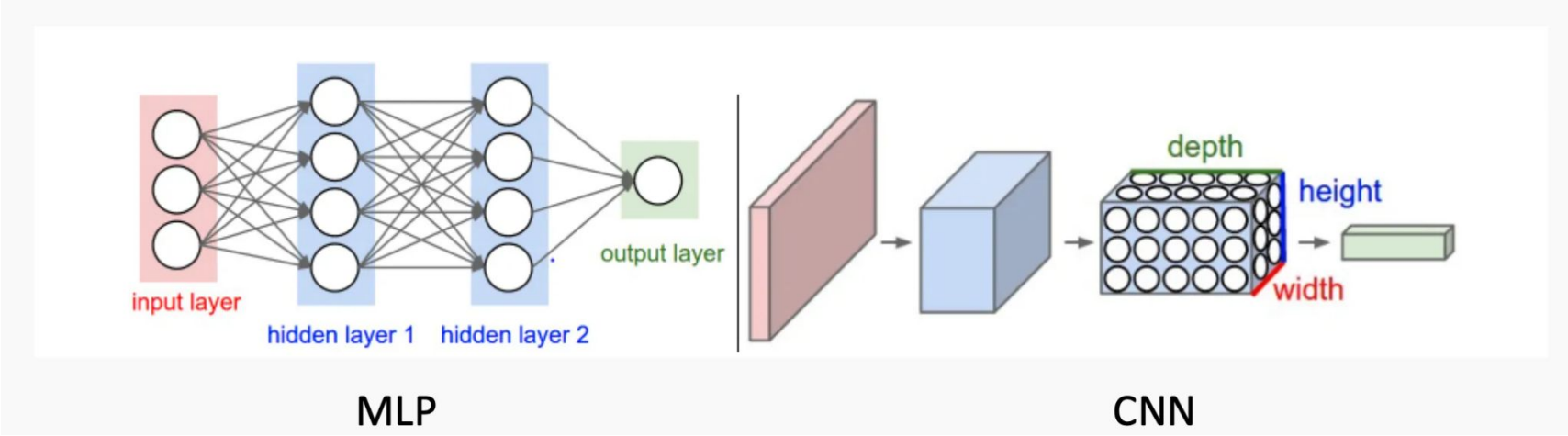
```
1/1 [==============================] - 1s 950ms/step
[[('n01806143', 'peacock', 0.9995116), ('n02017213', 'European_gallinule', 0.00019161004), ('n01847000', 'drake', 0.00011351371), ('n015
14668', 'cock', 3.6598827e-05), ('n01795545', 'black_grouse', 3.6351572e-05)]]
```

Active class

CNN con Keras

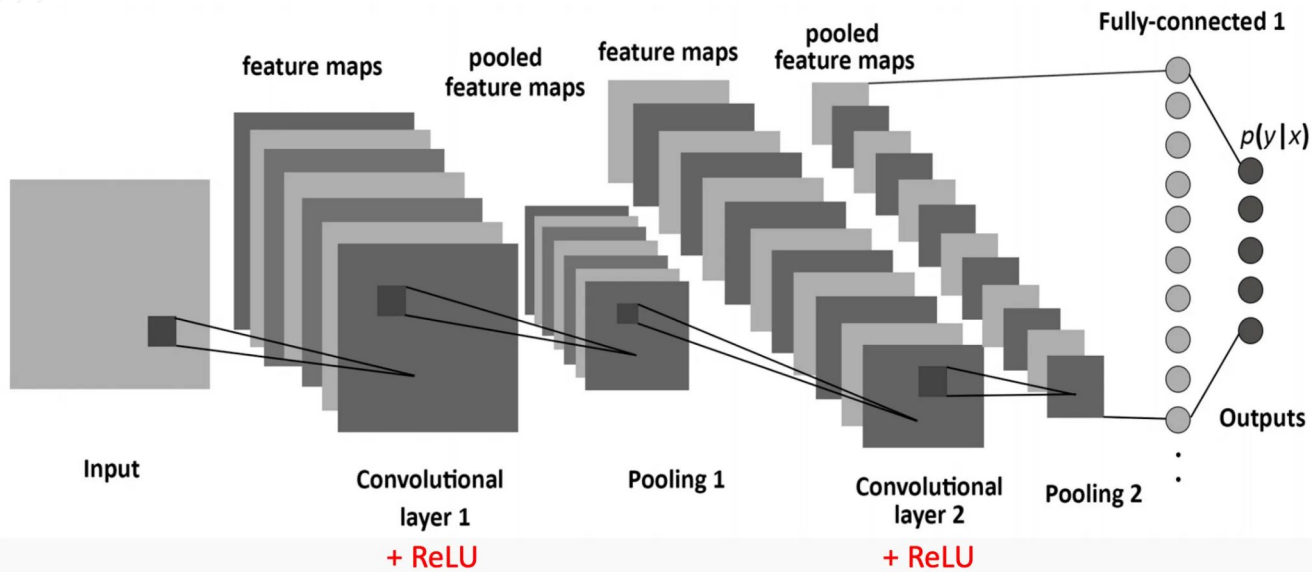
CNN

Al utilizar MLP para procesamiento de imágenes, es necesario representar la imagen como un vector y aplicar un pixel a cada nodo. La relación espacial que guardan los pixeles entre sí se pierde.



https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

CNN



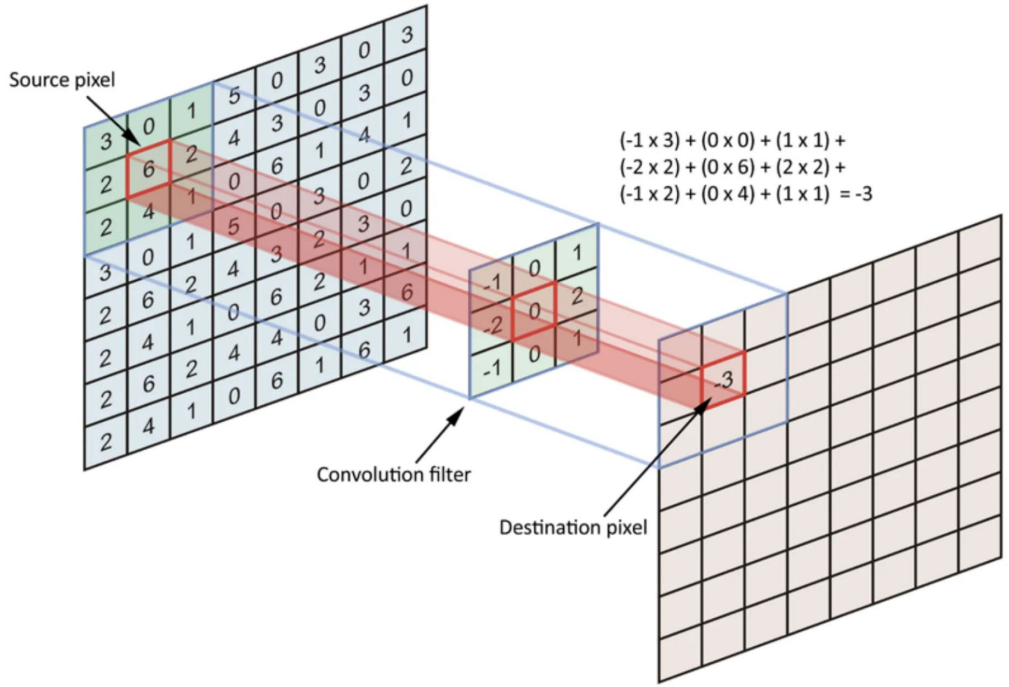
Las CNN están orientadas a extraer información espacial en imágenes y video a través de filtros bidimensionales.

Existen tres tipos de capas en una CNN:

- Convolutional
- Pooling
- Fully-connected

https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

CNN

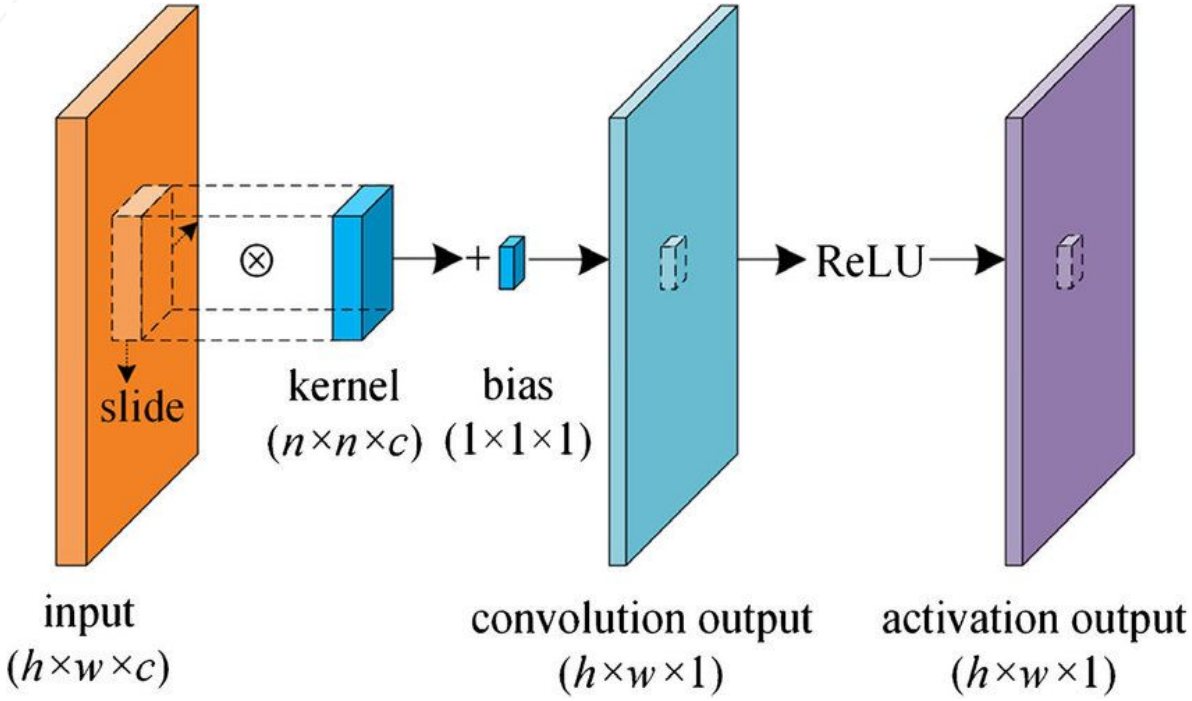


La capa Convolutional se basa en la operación de convolución y un kernel 2D, también llamado filtro. Los filtros se aplican para extraer características y crear mapas de características (feature maps). Los parámetros son

- Número de kernels
- Tamaño de kernels
- Función de activación
- Stride
- Padding

https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

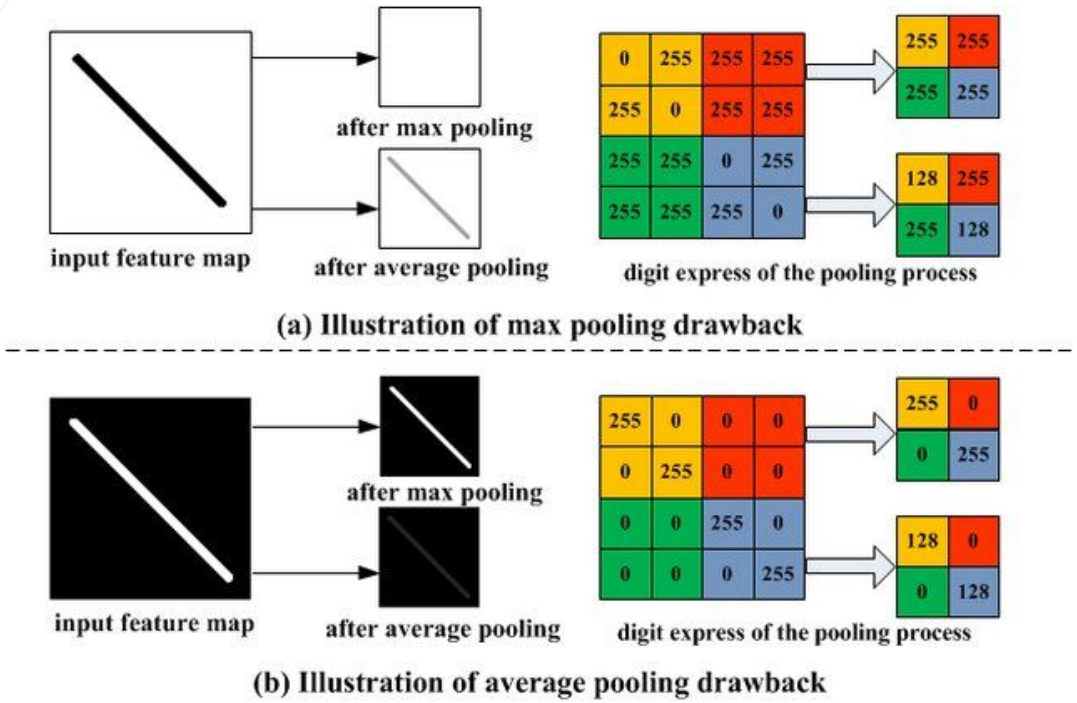
CNN



https://www.researchgate.net/figure/The-convolution-operation-in-the-CNN-The-convolutional-kernel-W-is-applied-on-the-input_fig1_338978963

La imagen muestra la aplicación del kernel y del bias a una imagen de entrada en una red CNN. Esta capa haciendo uso de la función de activación ReLU. Nótese que la imagen de entrada es de ***c*** canales, mientras que el ***feature map*** es de un solo canal.

CNN

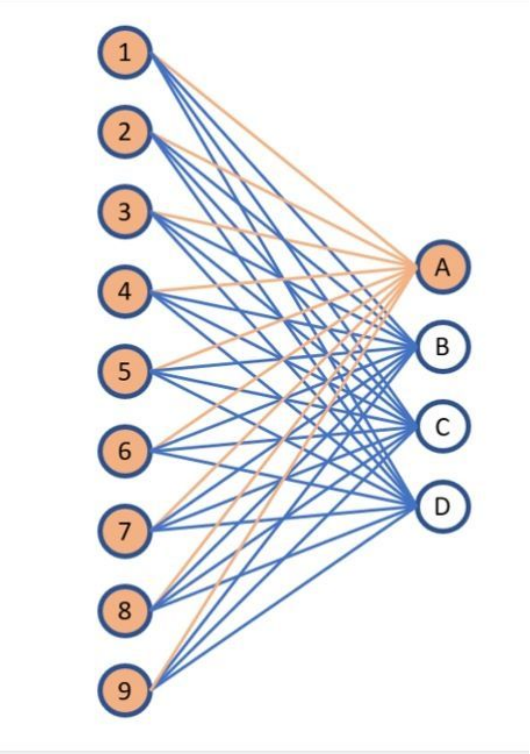


https://www.researchgate.net/publication/300020038_Mixed_Pooling_for_Convolutional_Neural_Networks/figures?lo=1

Las capas Pooling usualmente proceden de las capas Convolutional y reducen la dimensionalidad de la información recorriendo los features maps. En la imagen se muestran MaxPooling y AveragePooling. Los parámetros son

- Stride
- Tamaño de la ventana

CNN



https://builtin.com/machine-learning/fully-connected-layer

Las capas capas Fully-connected usualmente preceden a la salida de la clasificación y su objetivo es aplanar (flatten) los resultados. Estas son las capas de los MLP. Los parámetros son

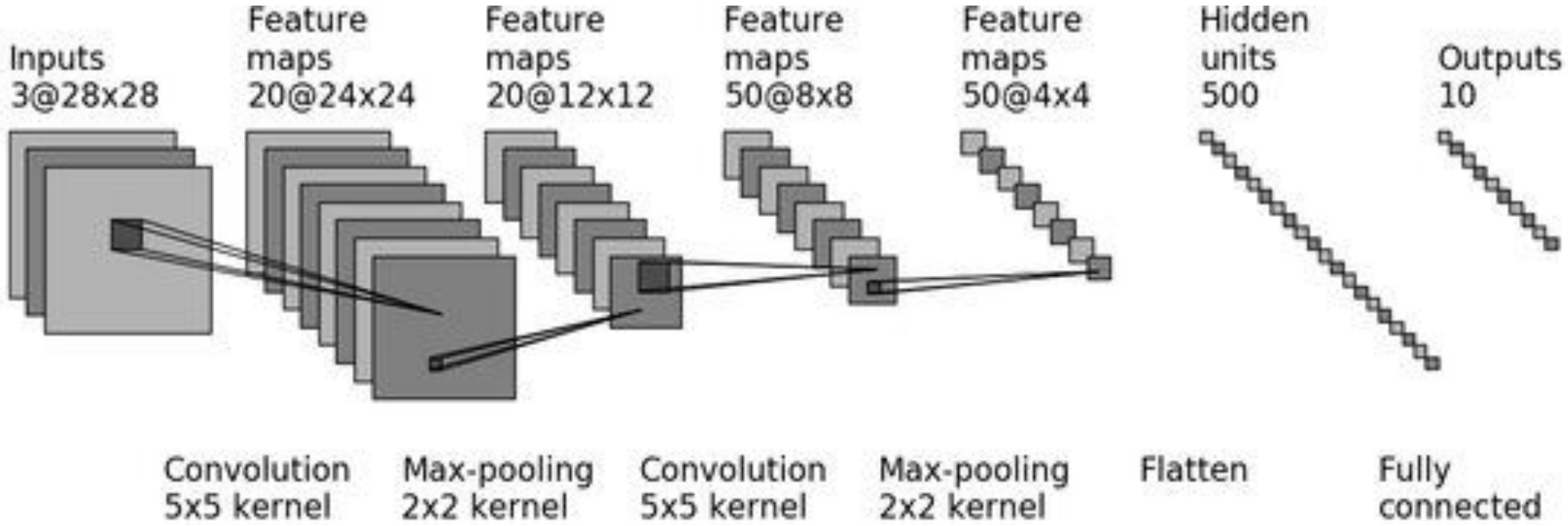
- Número de nodos
- Función de activación

CNN

```python
model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (3, 3), input_shape = X_train.shape[1:], activation='relu', padding='same'))
model.add(Conv2D(filters = 32, kernel_size = (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
#model.add(AveragePooling2D())

model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation='relu', padding='same'))
model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation='relu', padding='same'))
model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(units = 512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 10, activation='softmax'))

model.compile(Adam(lr = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])
print(model.summary())
```

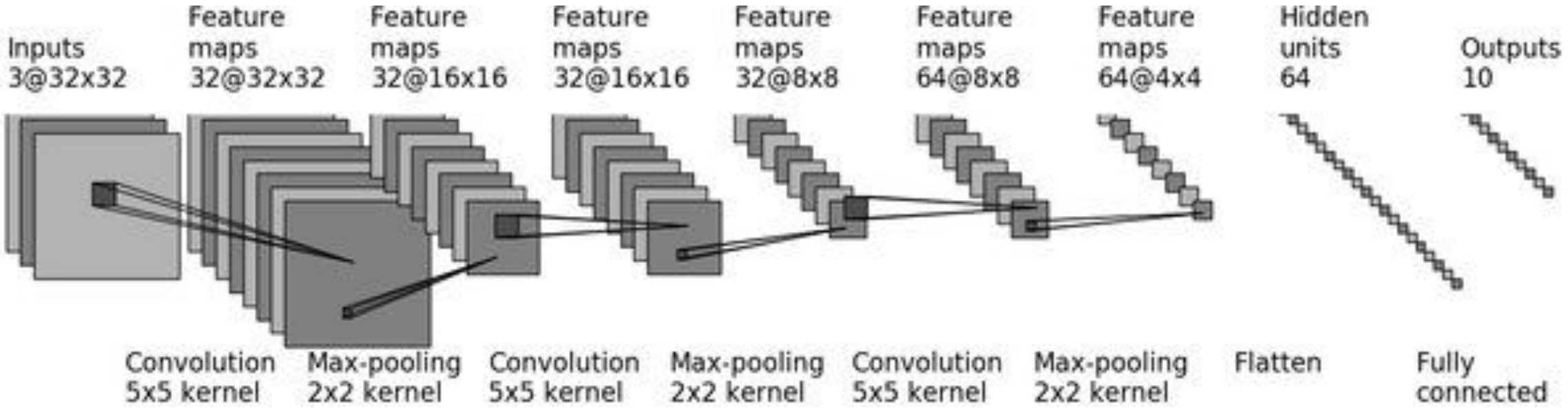
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 conv2d_1 (Conv2D)           (None, 30, 30, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 dropout (Dropout)           (None, 15, 15, 32)        0

 conv2d_2 (Conv2D)           (None, 15, 15, 64)        18496

 conv2d_3 (Conv2D)           (None, 13, 13, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 6, 6, 64)          0

 conv2d_4 (Conv2D)           (None, 6, 6, 64)          36928

 conv2d_5 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 dropout_2 (Dropout)         (None, 2, 2, 64)          0

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 512)               131584

 dropout_3 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 276,138
Trainable params: 276,138
Non-trainable params: 0
```

CNN - LeNet5



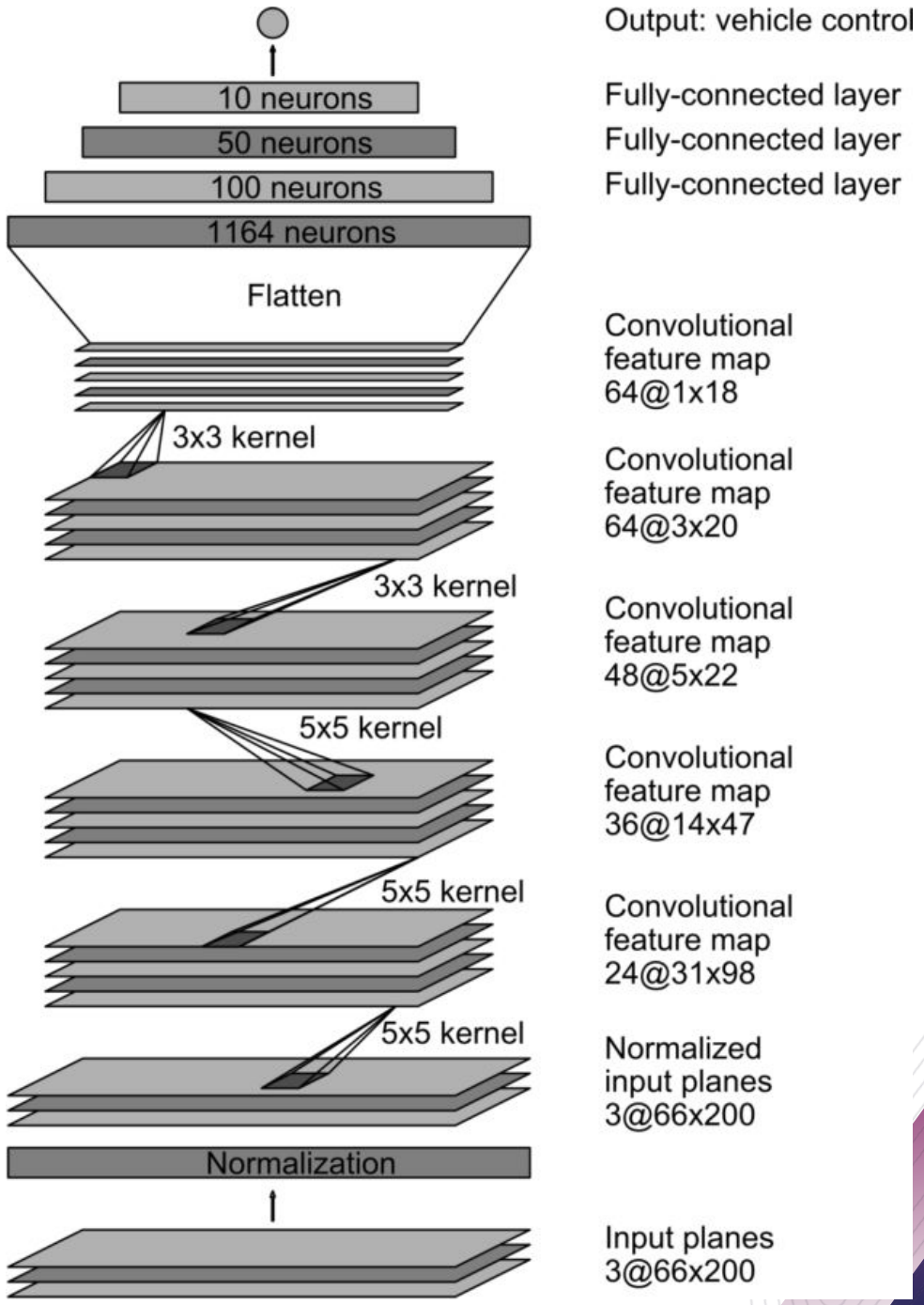
https://www.researchgate.net/publication/312170477_On_Classification_of_Distorted_Images_with_Deep_Convolutional_Neural_Networks/figures?lo=1

CNN - CIFAR10



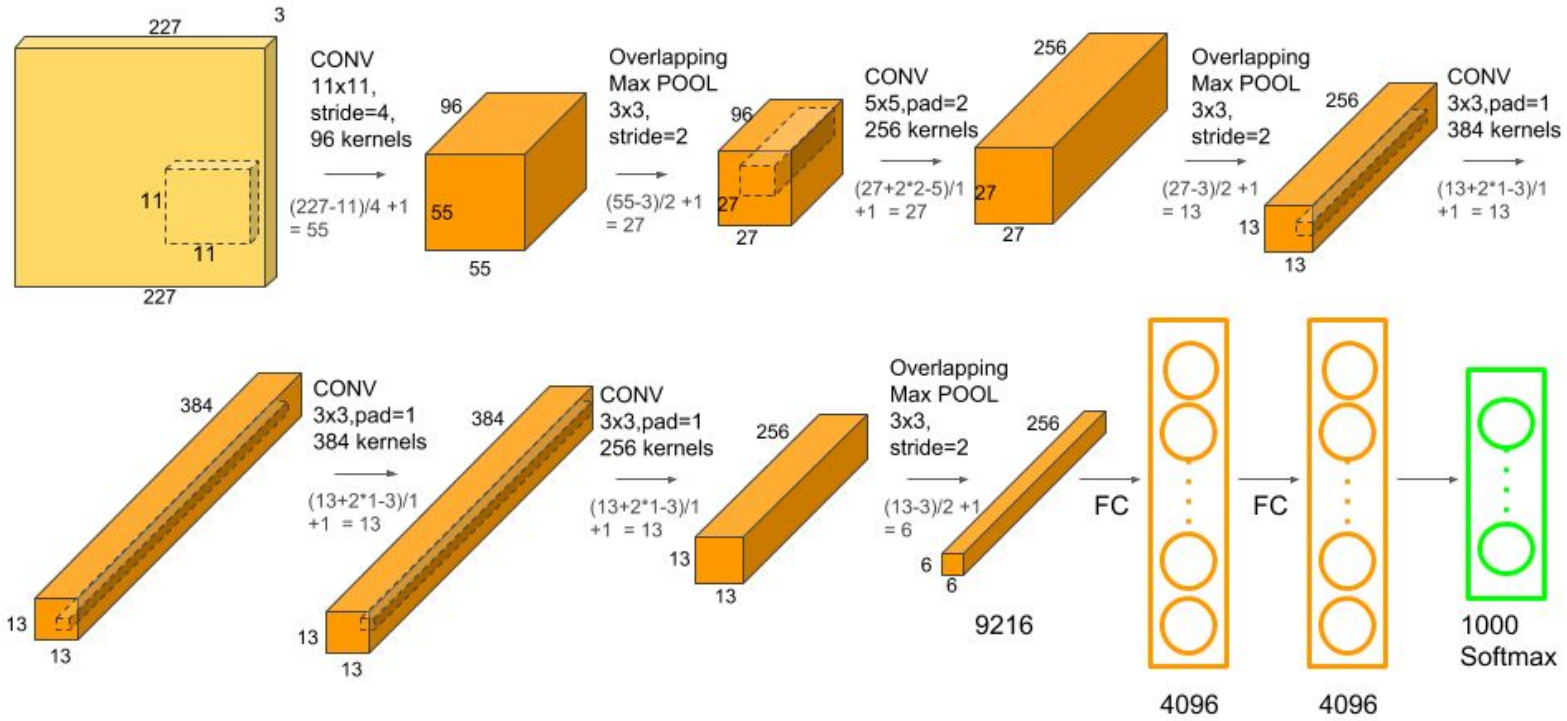
https://www.researchgate.net/publication/312170477_On_Classification_of_Distorted_Images_with_Deep_Convolutional_Neural_Networks/figures?lo=1

CNN - Nvidia Model



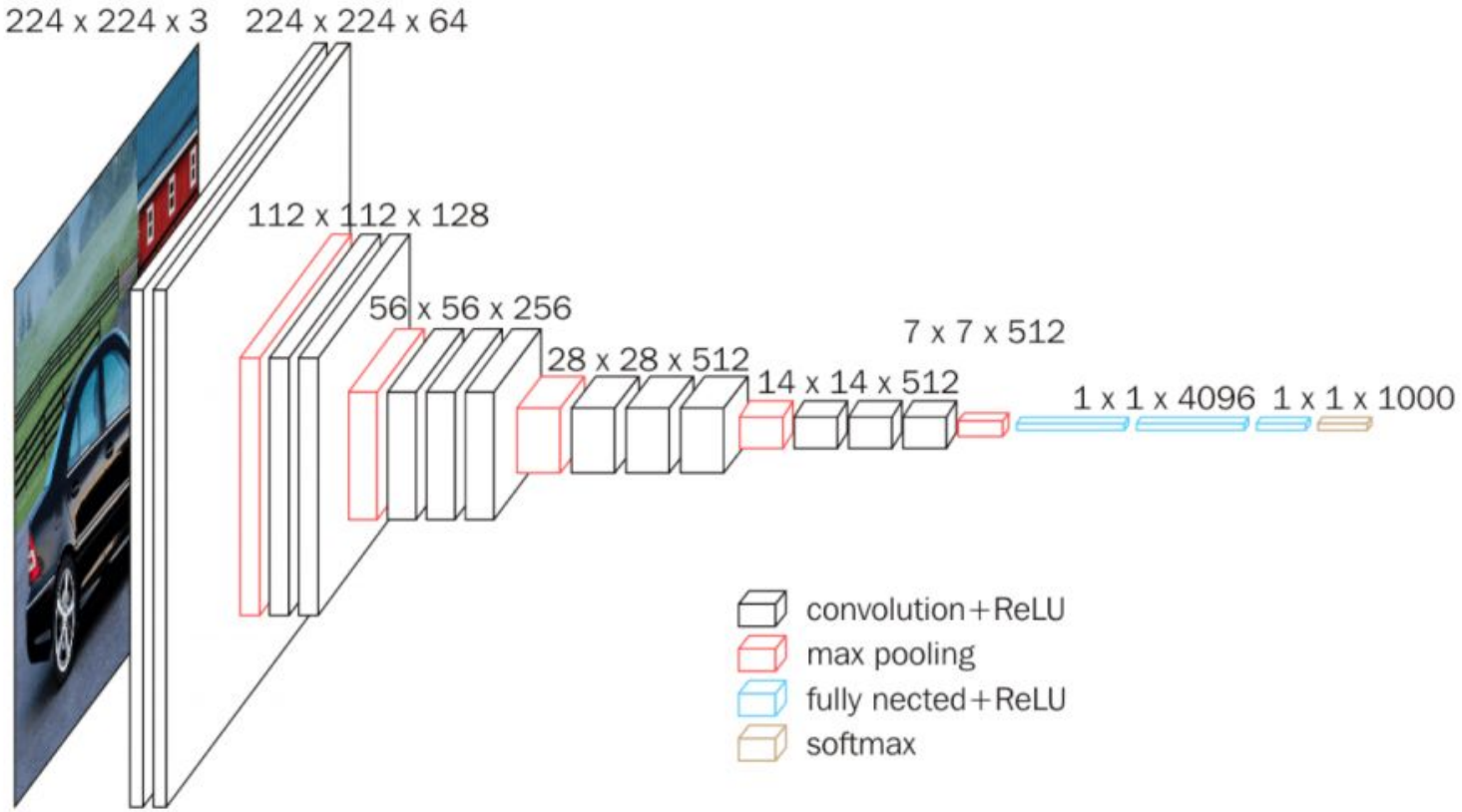
https://developer.nvidia.com/blog/deep-learning-self-driving-cars/

CNN - AlexNet



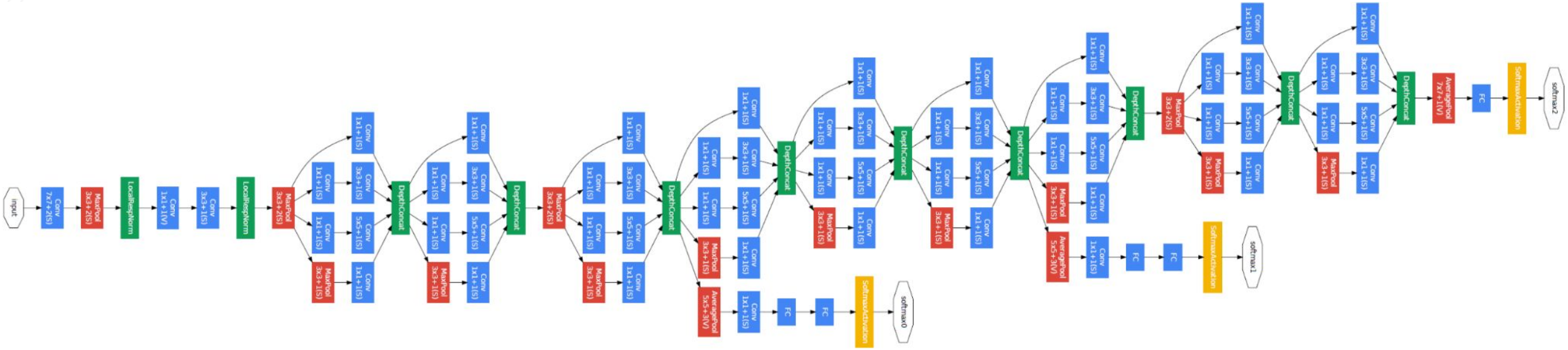
https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-vgg-googlenet/

CNN - VGG16



https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-vgg-googlenet/

CNN - GoogLeNet



https://arxiv.org/pdf/1409.4842.pdf

Active class

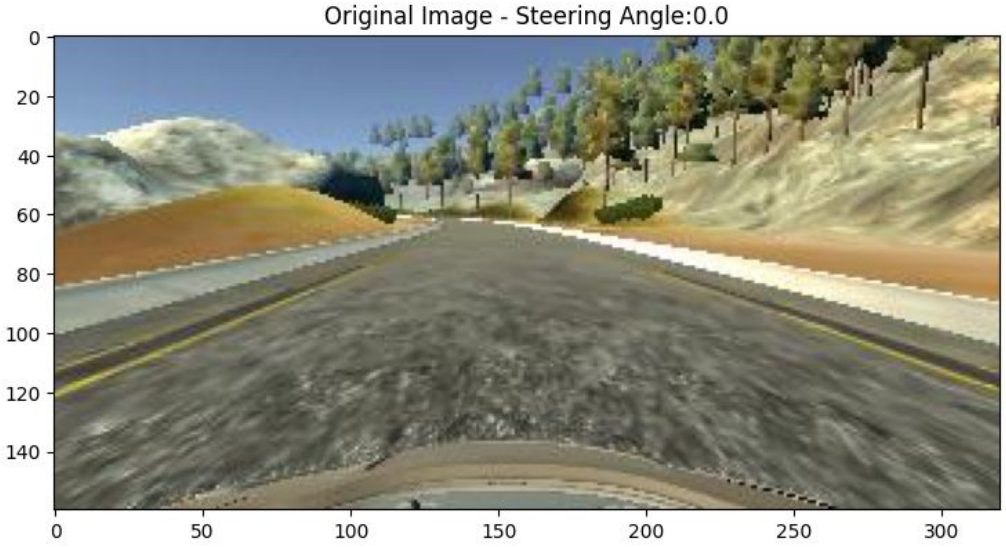
Data Augmentation

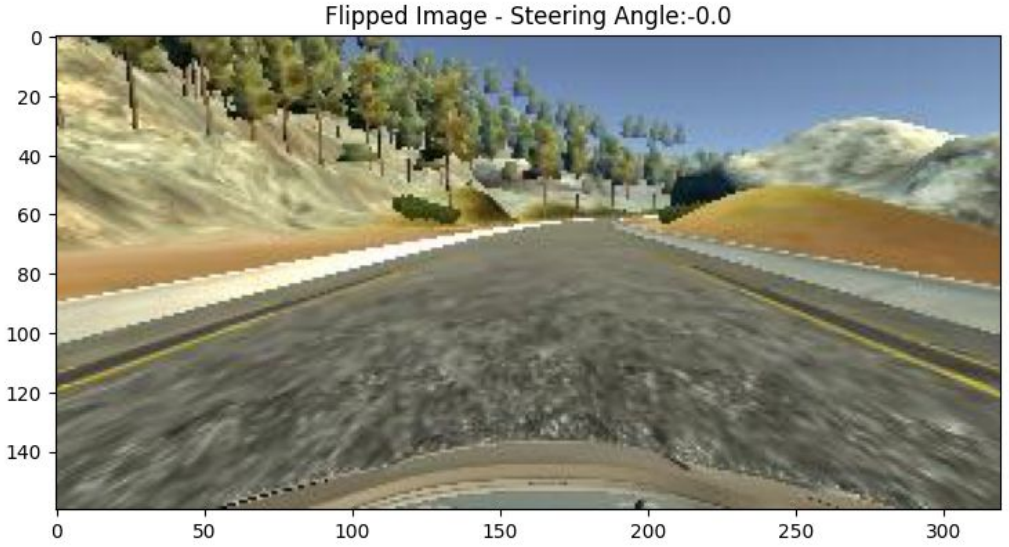
Data Augmentation

https://imgaug.readthedocs.io/en/latest/

https://towardsdatascience.com/image-augmentation-with-skimage-python-7f53f1b0eeb5

Es el proceso de las muestras en el dataset creando nuevas imágenes a partir de las ya existentes. Se puede usar ***ImageDataGenerator*** de Keras, ***Scikit-image*** o la biblioteca ***imgaug***. Rotación, zoom, brightness, flip, etc. son algunas operaciones adecuadas para este paso.





Active class

Referencias

Referencias

- https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141
- https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/
- https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32
- https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
- https://keras.io/api/applications/
- https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac
- https://www.researchgate.net/publication/300020038_Mixed_Pooling_for_Convolutional_Neural_Networks/figures?lo=1
- https://www.researchgate.net/publication/312170477_On_Classification_of_Distorted_Images_with_Deep_Convolutional_Neural_Networks/figures?lo=1
- https://developer.nvidia.com/blog/deep-learning-self-driving-cars/
- https://arxiv.org/pdf/1409.4842.pdf
- https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-vgg-googlenet/

Tecnológico de Monterrey

D.R.© Tecnológico de Monterrey, México, 2023.
Prohibida la reproducción total o parcial
de esta obra sin expresa autorización del
Tecnológico de Monterrey.