Agenda

- **anuncios varios**
  - Tarea 2 entrega lunes 8 de Mayo (Preguntas)
- **modelos de analitica (machine learning-ML) Supervisado**
  - **Redes Neuronales**
    - **NNets**
      - **Dropout**
    - **Convolutional Neural Net**
      - **Operador convolución**
      - **Data augmentation**
    - **LSTM**
      - **Series de tiempo**
      - **Secuencias**

Redes Neuronales

Redes Neuronales: Es un conjunto de combinaciones lineales de los features con una capa de activación no lineal. De esa forma cada neurona aprende combinaciones diferentes que pueden ser combinadas nuevamente con una segunda capa. Dos capas pueden aprender cualquier función continua.

$$\begin{aligned} f(X) &= \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) \\ &= \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j). \end{aligned}$$

$$A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j),$$

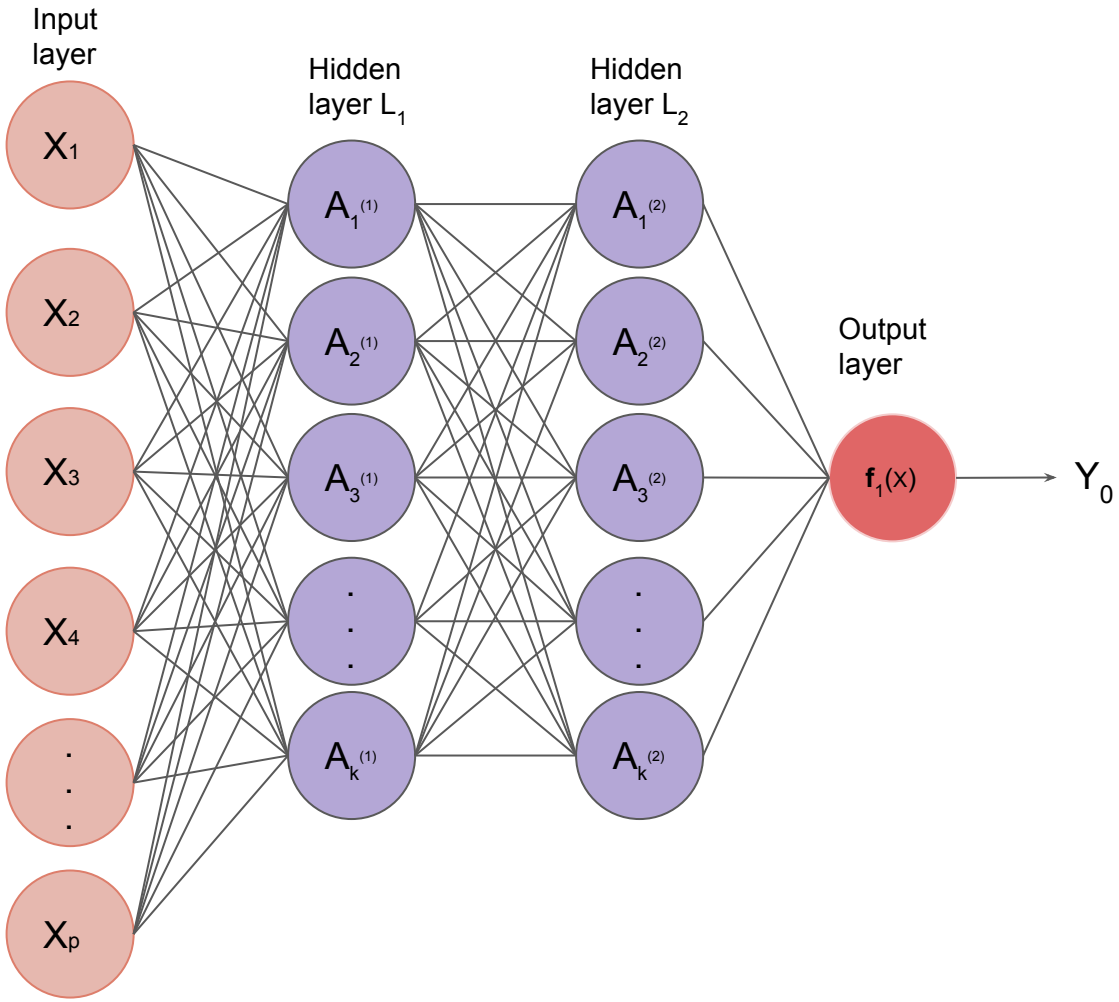
$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k,$$

K es el numero de Neuronas por cada capa

* A Course of Machine Learning http://ciml.info/

Redes Neuronales, ejemplo modelo dataset de carros

Por ejemplo, para predecir si el carro es amigable con el medio ambiente o no. Podemos utilizar 2 capas



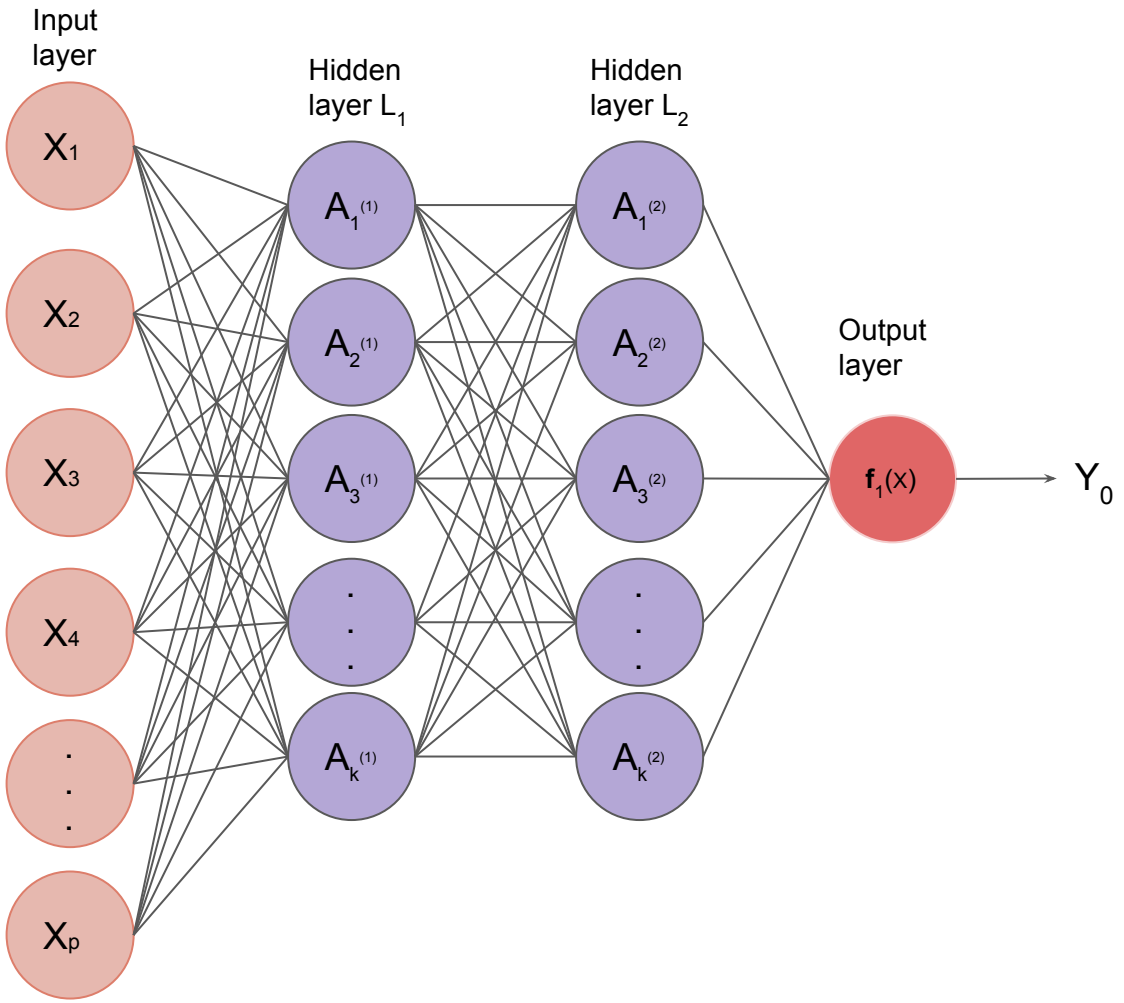
Función de costo o error Binary Cross Entropy

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

* A Course of Machine Learning http://ciml.info/

Redes Neuronales, ejemplo modelo dataset de carros

Arquitectura



Python code

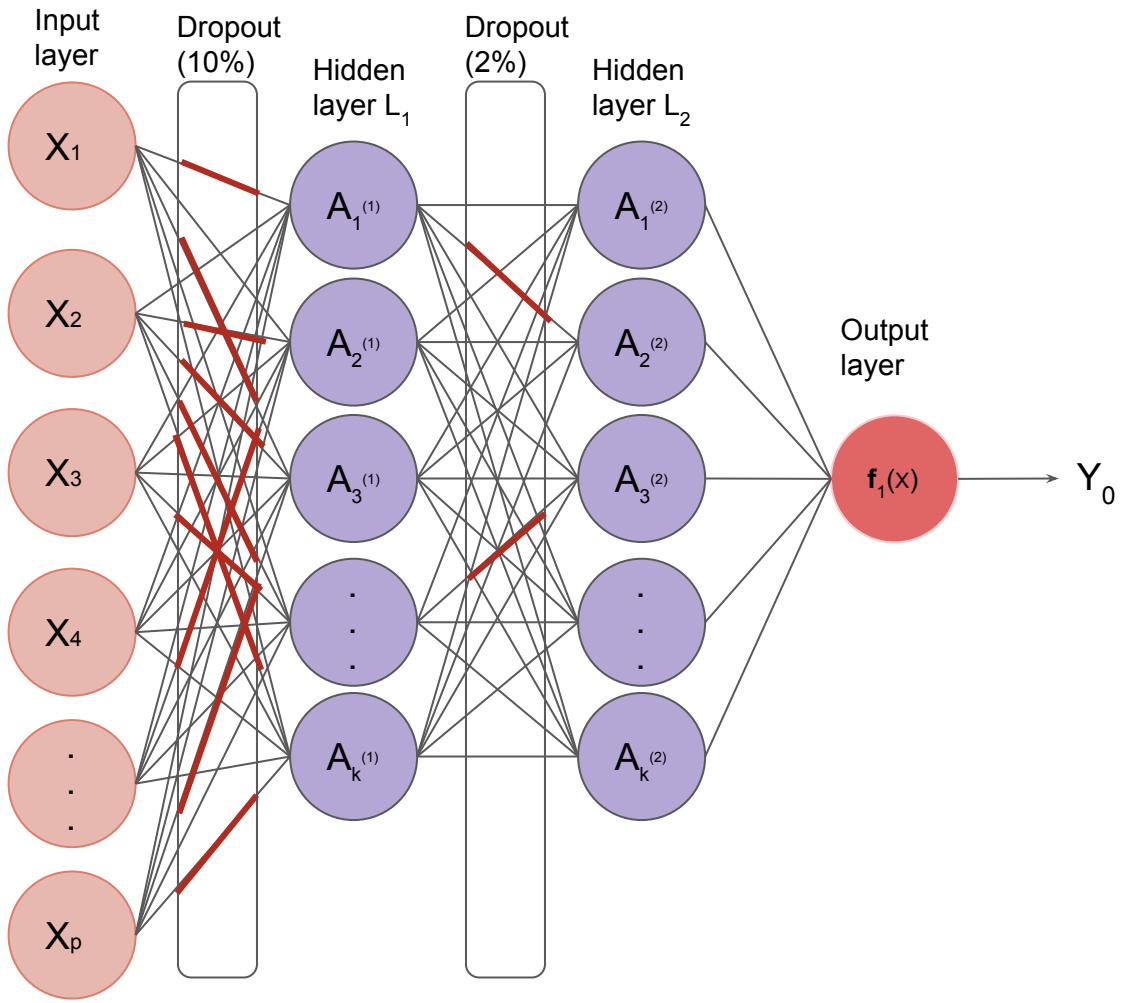
```
# first neural network with keras tutorial
from numpy import loadtxt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(20, input_shape=(20,)))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

* A Course of Machine Learning http://ciml.info/

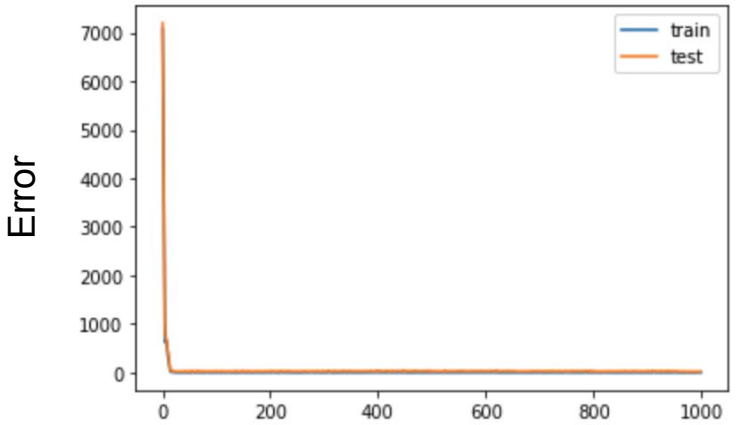
Redes Neuronales, ejemplo carros con dropout

Dropout es destruir conecciones aleatoriamente para reducir el overfitting



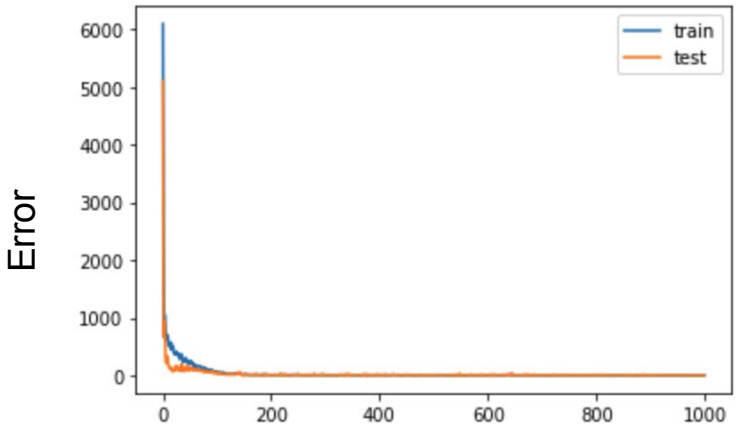
Resultado sin Dropout

```
Train: 0.886, Test: 0.767
```



Resultado con Dropout

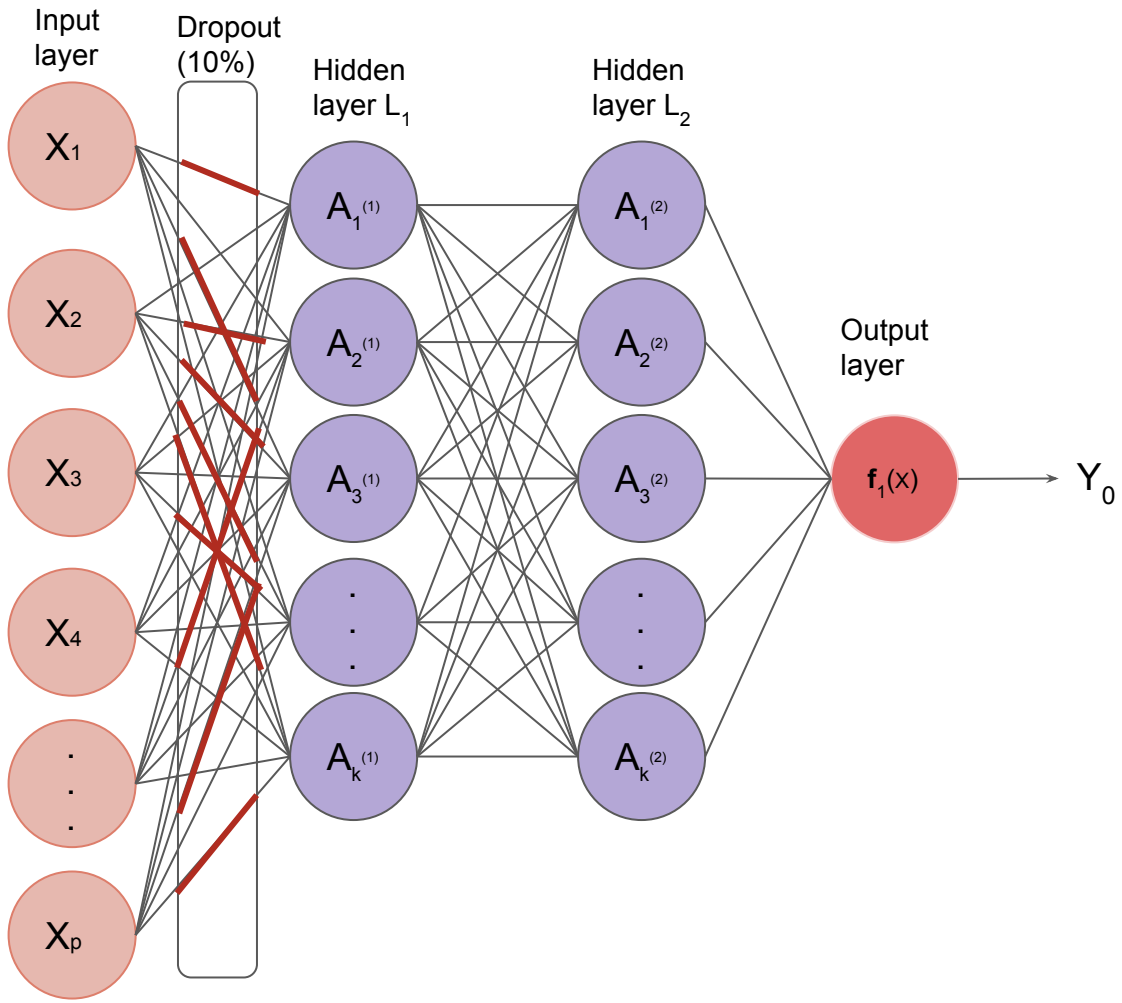
```
Train: 0.886, Test: 0.814
```



* A Course of Machine Learning http://ciml.info/

Redes Neuronales, ejemplo carros con dropout

Dropout es destruir conexiones aleatoriamente para reducir el overfitting

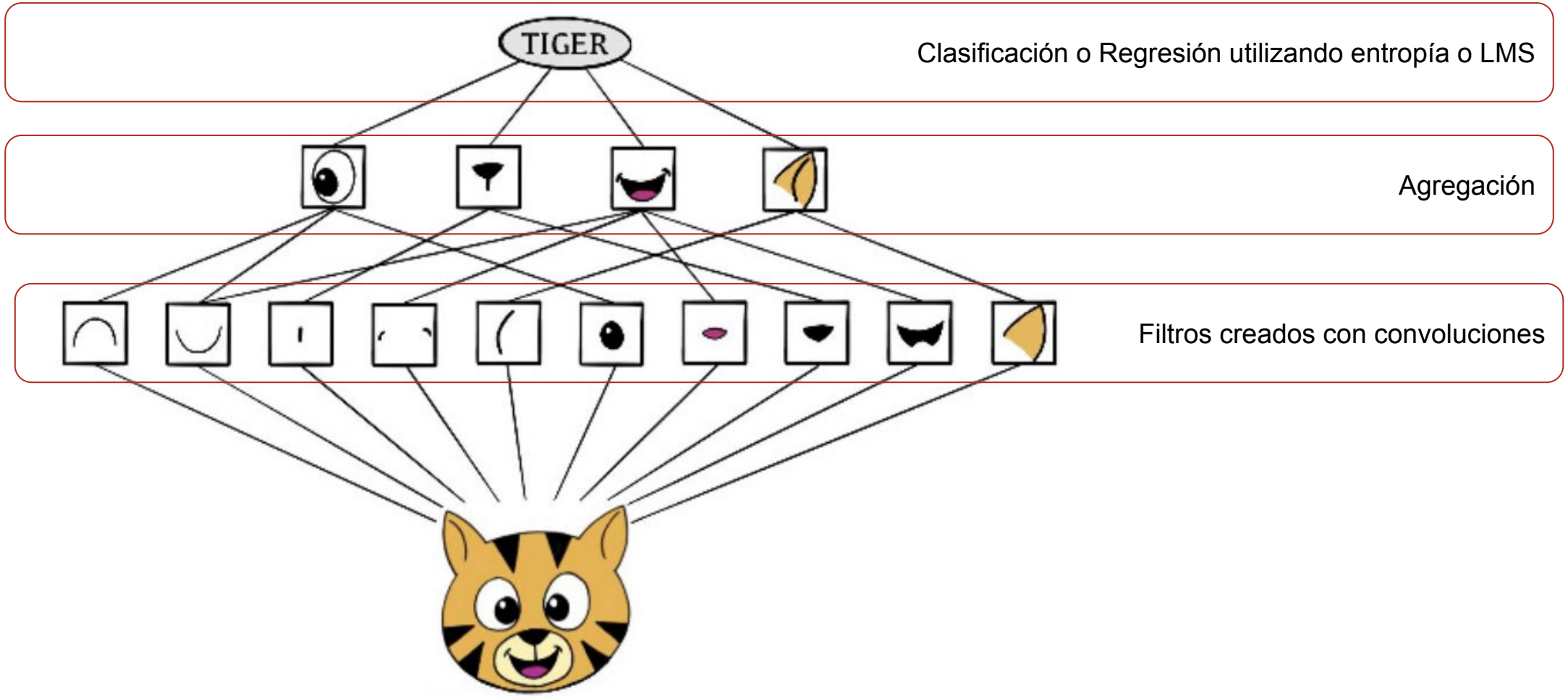



```
model = Sequential()
model.add(Dense(10, input_shape=(20,)))
model.add(Dropout(0.1))
model.add(Dense(5, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

* A Course of Machine Learning http://ciml.info/

Redes Neuronales Convolucionales

CNNs mimic to some degree how humans classify images, by recognizing specific features or patterns anywhere in the image



Krizhevsky (2009) "Learning multiple layers of features from tiny images", available at https://www.cs.toronto.edu/~kriz/ learning-features-2009-TR.pdf

* A Course of Machine Learning http://ciml.info/

Redes Neuronales Que es una convolución.

La operación convolución en 2D. Esto se repite por cada color RGB y es llamado canales

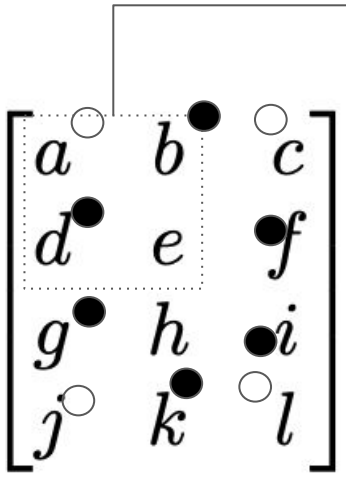
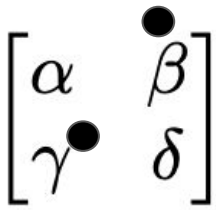


Image 4 x 3



Convolution filter 2 x 2

$$\begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

Resultado de la convolución

$$\forall i \in [C'], \mathrm{Conv2D}(z)_i = \sum_{j=1}^{C} \mathrm{Conv2D\text{-}S}_{i,j}(z_j).$$

Krizhevsky (2009) “Learning multiple layers of features from tiny images”, available at https://www.cs.toronto.edu/~kriz/ learning-features-2009-TR.pdf

* A Course of Machine Learning http://ciml.info/

Redes Neuronales Pooling

La operación de hacer sampling de una matriz y represéntalo con el promedio o el máximo valor.

$$\begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

Pooling 2 x 2

Convolution result

$$\text{MAX} \begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \end{bmatrix}$$

$$\text{MAX} \begin{bmatrix} d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

Resultado Max pooling or Avg pooling

$$\text{Max pool} \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}.$$

Krizhevsky (2009) “Learning multiple layers of features from tiny images”, available at https://www.cs.toronto.edu/~kriz/ learning-features-2009-TR.pdf

* A Course of Machine Learning http://ciml.info/

Redes Neuronales Convolucionales, historia

LeNet fue creada por Yann LeCun entre 1989-1998 para reconocer los números

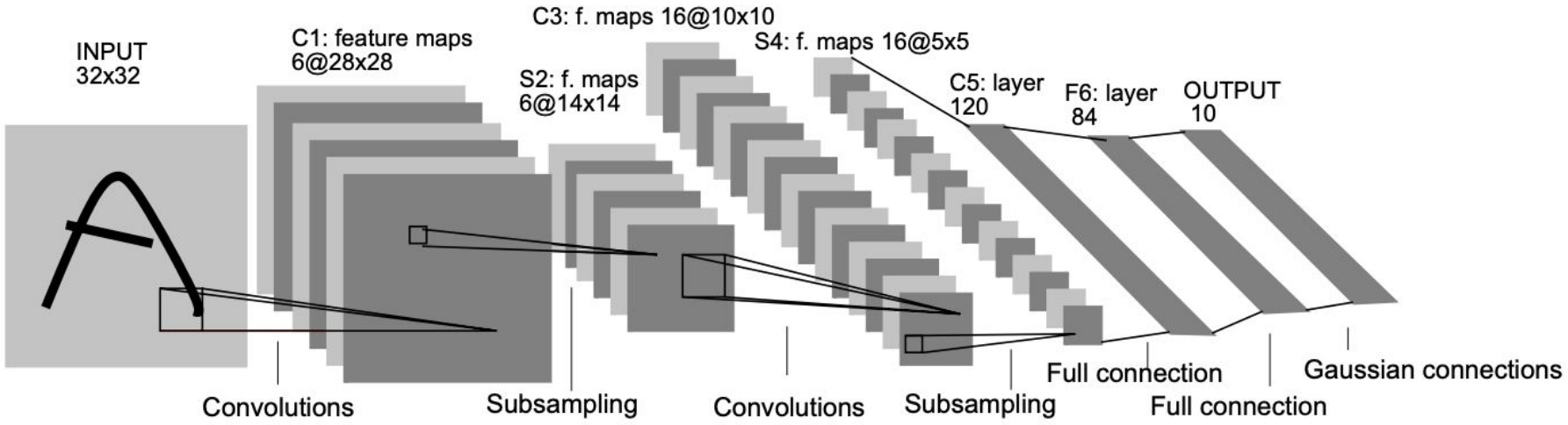


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

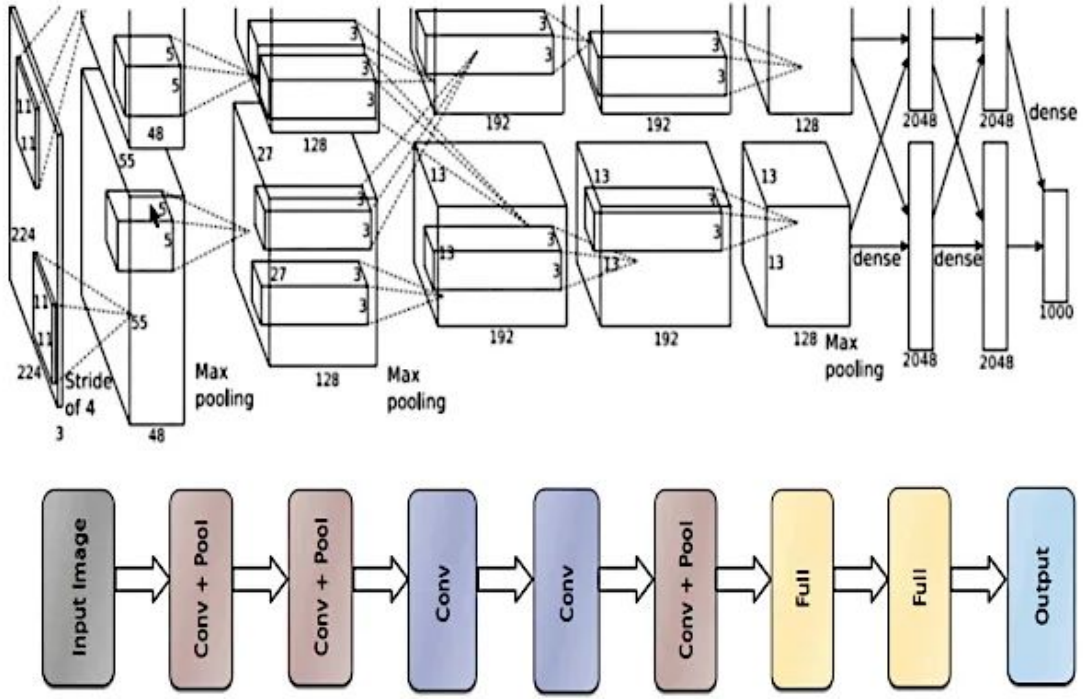
http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

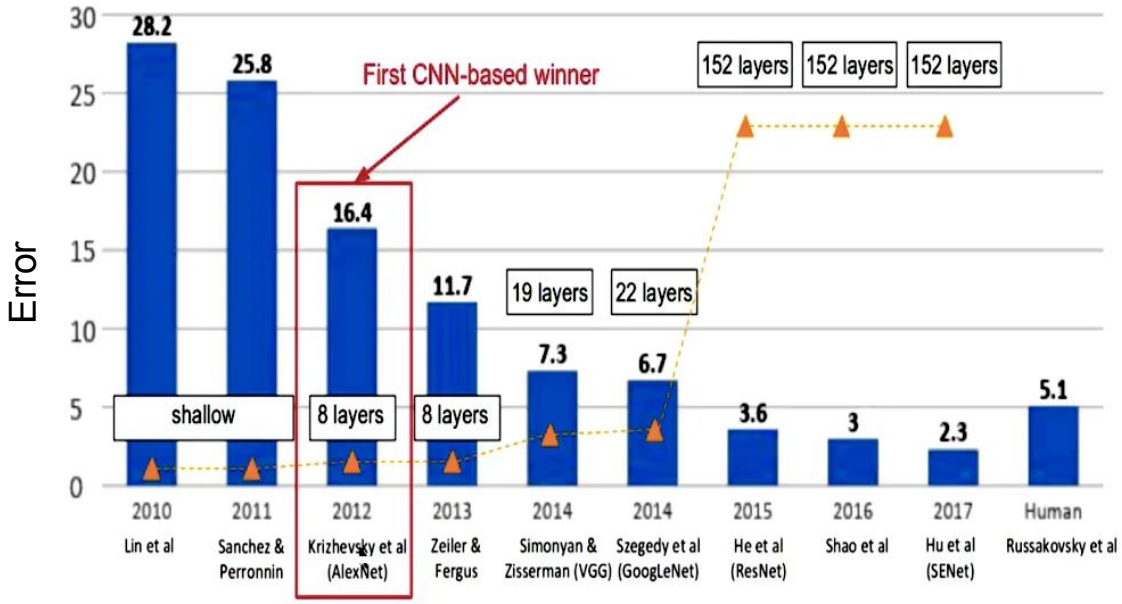
Krizhevsky (2009) “Learning multiple layers of features from tiny images”, available at https://www.cs.toronto.edu/~kriz/ learning-features-2009-TR.pdf

* A Course of Machine Learning http://ciml.info/

Redes Neuronales Convolucionales, historia

AlexNet (2012) fue el primer ganador del Imagenet challenge (data set con 150,000 fotos y 1000 categorías) utilizando 8 layers.





http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

Krizhevsky (2009) “Learning multiple layers of features from tiny images”, available at https://www.cs.toronto.edu/~kriz/ learning-features-2009-TR.pdf

* A Course of Machine Learning http://ciml.info/

Redes Neuronales Convolucionales,

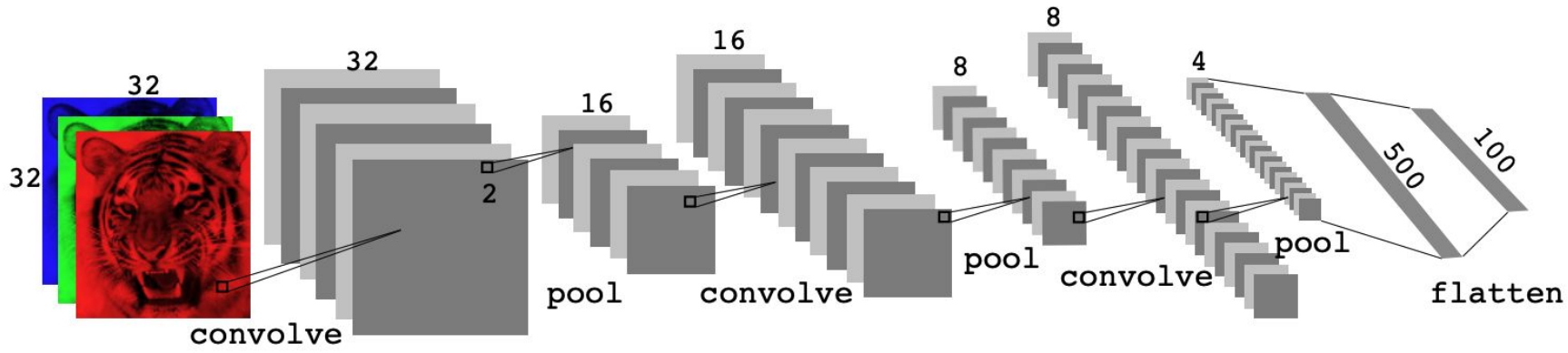


FIGURE 10.8. *Architecture of a deep CNN for the* CIFAR100 *classification task. Convolution layers are interspersed with* 2×2 *max-pool layers, which reduce the size by a factor of 2 in both dimensions.*

http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

Krizhevsky (2009) “Learning multiple layers of features from tiny images”, available at https://www.cs.toronto.edu/~kriz/ learning-features-2009-TR.pdf

* Gareth James, An Introduction to Statistical Learning

Redes Neuronales Convolucionales, Data Augmentation

Mejorar el overfitting del modelo utilizando técnicas para mejorar la clasificación de imágenes como:

- Rotación en X, Y
- Adicionar ruido
- Mover a la izquierda
- Mover a la derecha
- Invertir las imágenes

FIGURE 10.9. *Data augmentation. The original image (leftmost) is distorted in natural ways to produce different images with the same class label. These distortions do not fool humans, and act as a form of regularization when fitting the CNN.*

http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

Krizhevsky (2009) “Learning multiple layers of features from tiny images”, available at https://www.cs.toronto.edu/~kriz/ learning-features-2009-TR.pdf

* Gareth James, An Introduction to Statistical Learning