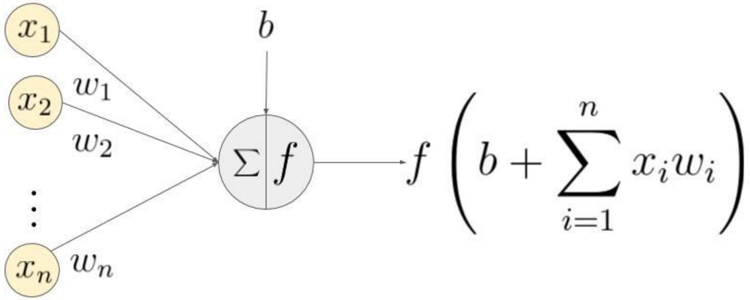
Artificial Neural Networks

Applications of neural networks

- Natural Language processing (text classification, summarization, topic modelling, etc)
- **Image Classification**
- Object Detection
- Time series forecasting
- Speech recognition
- Recommendation systems
- Synthetic image and signals generation
- Dimensionality reduction and denoising

Let’s start by a simple perceptron

Feed forward view: Inputs are introduced into the networks to produce a result.



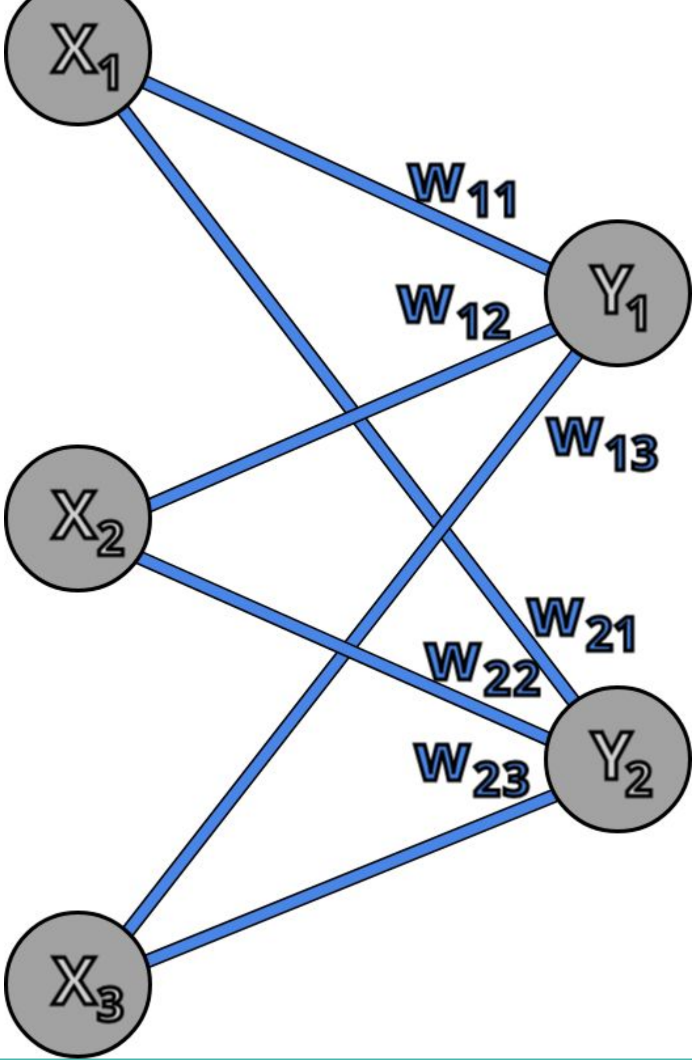
- x1, x2, …, xn are feature variables
- w1, w2, …, wn are the weights or the connections from features to the output
- b is bias
- f is called an activation function. It will allow non linear relationships

Then move on to a fully connected perceptron

$$Y_2 = \sigma(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_2)$$

$$Y_2 = \sigma(w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + b_2)$$

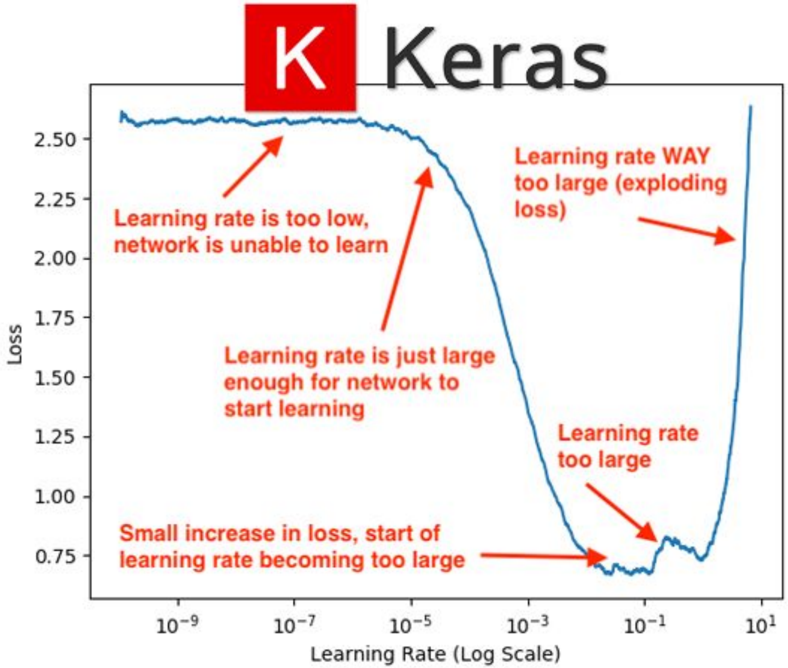
$$Y = \sigma(Wx + b)$$



Backpropagation

We would expect our outputs to have some degree of error. Our goal is to train the network to minimize it.

$$\Delta w_{kl}^{(m)} = -\eta \frac{\partial E(\{w_{ij}^{(n)}\})}{\partial w_{kl}^{(m)}}$$



Loss Functions

Mean Square Error (regression)

$$\mathrm{MSE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{test set}} (\underbrace{y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i}_{\text{actual value}})^2$$

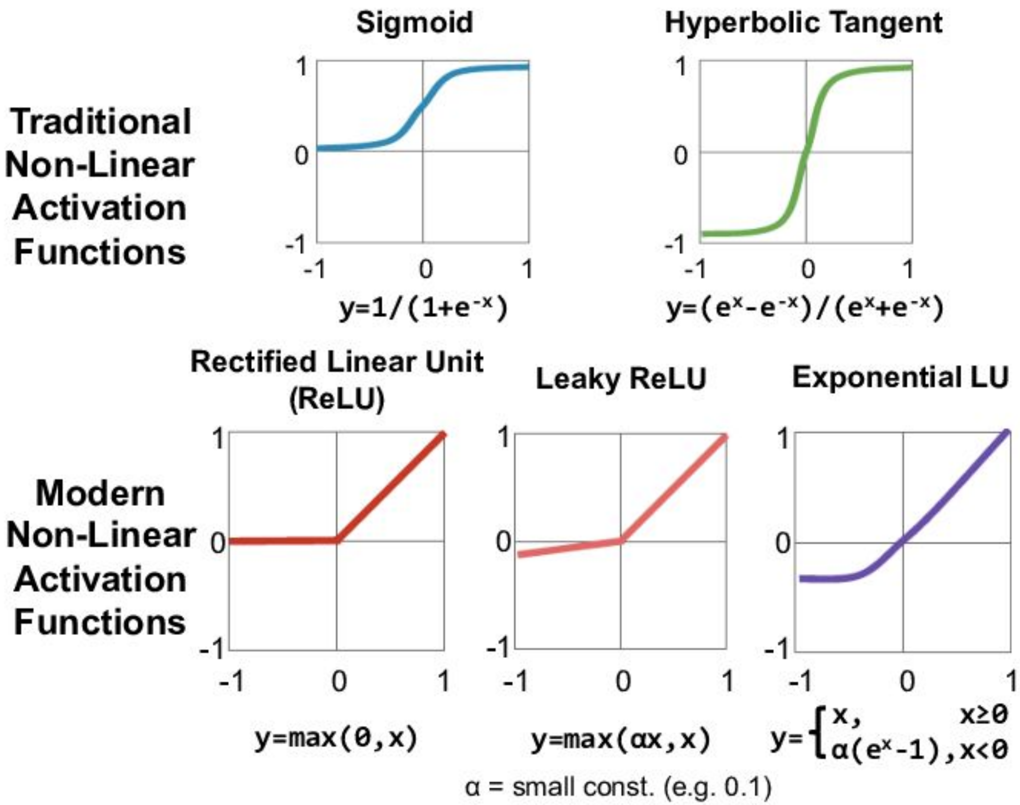
Categorical cross entropy Loss
(multi-class classification)

$$CE = -\sum_{i}^{C} t_i log(s_i)$$

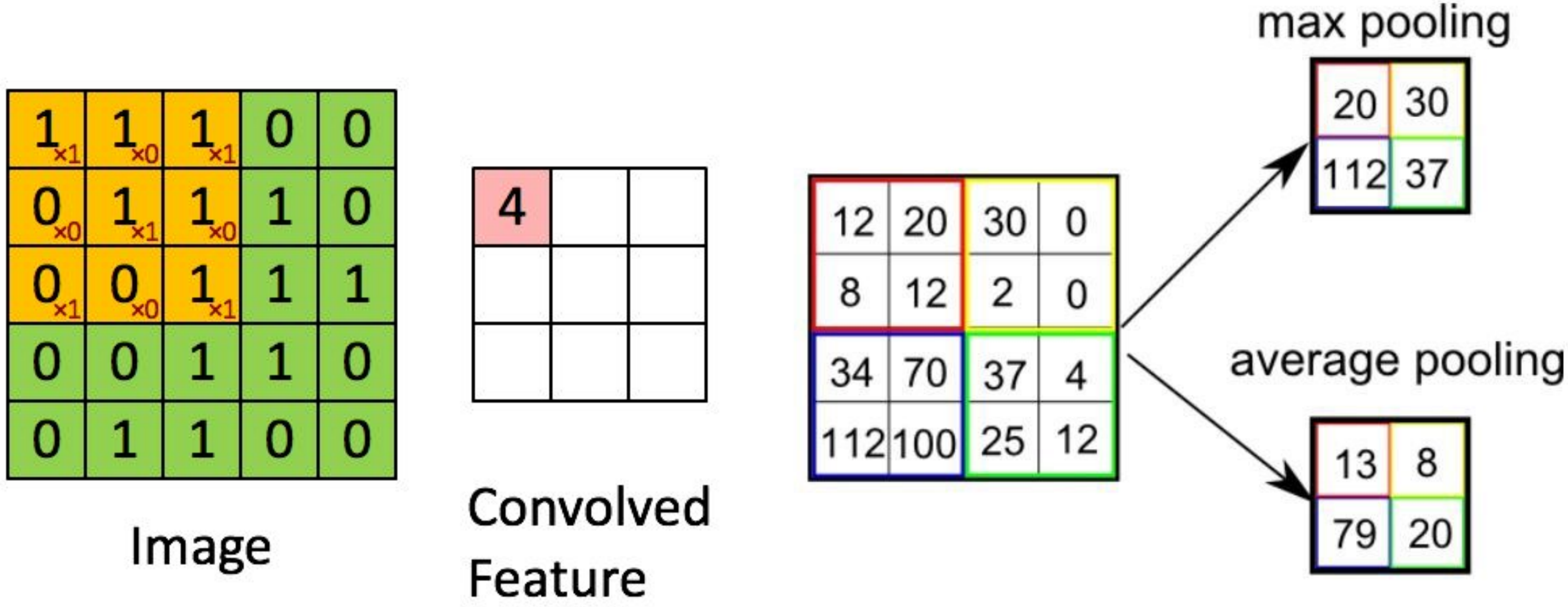
Binary Cross Entropy (Binary Classification)

$$CE = -\sum_{i=1}^{C'=2} t_i log(s_i) = -t_1 log(s_1) - (1 - t_1) log(1 - s_1)$$

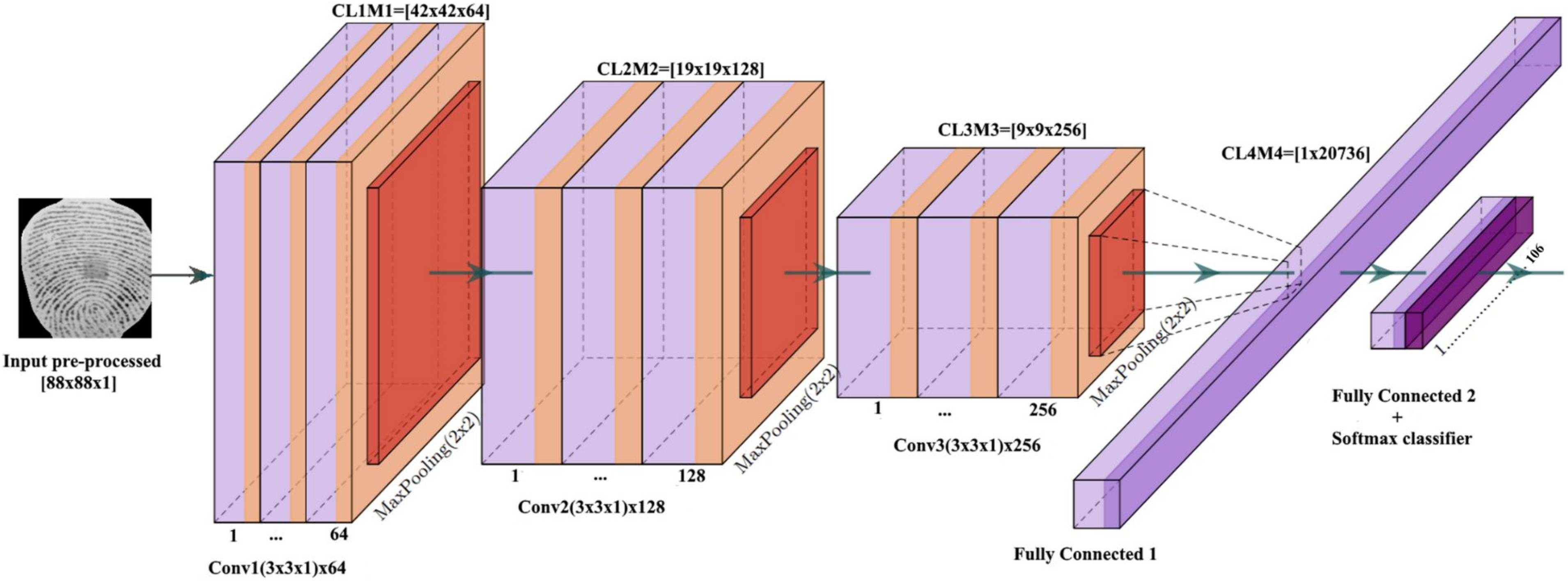
Activation Functions



Convolutional Filters and pooling layers



Basic Convolutional neural network



Keras Applications

Most popular architectures, ready to use in Keras

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
# Predicted: [(u'n02504013', u'Indian_elephant', 0.82658225), (u'n01871265', u'tusker', 0.1122
```

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
EfficientNetB0	29 MB	-	-	5,330,571	-
EfficientNetB1	31 MB	-	-	7,856,239	-
EfficientNetB2	36 MB	-	-	9,177,569	-
EfficientNetB3	48 MB	-	-	12,320,535	-
EfficientNetB4	75 MB	-	-	19,466,823	-
EfficientNetB5	118 MB	-	-	30,562,527	-
EfficientNetB6	166 MB	-	-	43,265,143	-
EfficientNetB7	256 MB	-	-	66,658,687	-

See more...

Deep Learning with TensorFlow | Free Courses in Data Science, AI, Cloud Computing, Containers, Kubernetes, Blockchain and more. (cognitiveclass.ai)

Deep Learning Fundamentals | Free Courses in Data Science, AI, Cloud Computing, Containers, Kubernetes, Blockchain and more. (cognitiveclass.ai)