Convolutional Neural Networks (CNNs)

Anahita Soltani Naveh

Mahdis Esmaeilzadeh

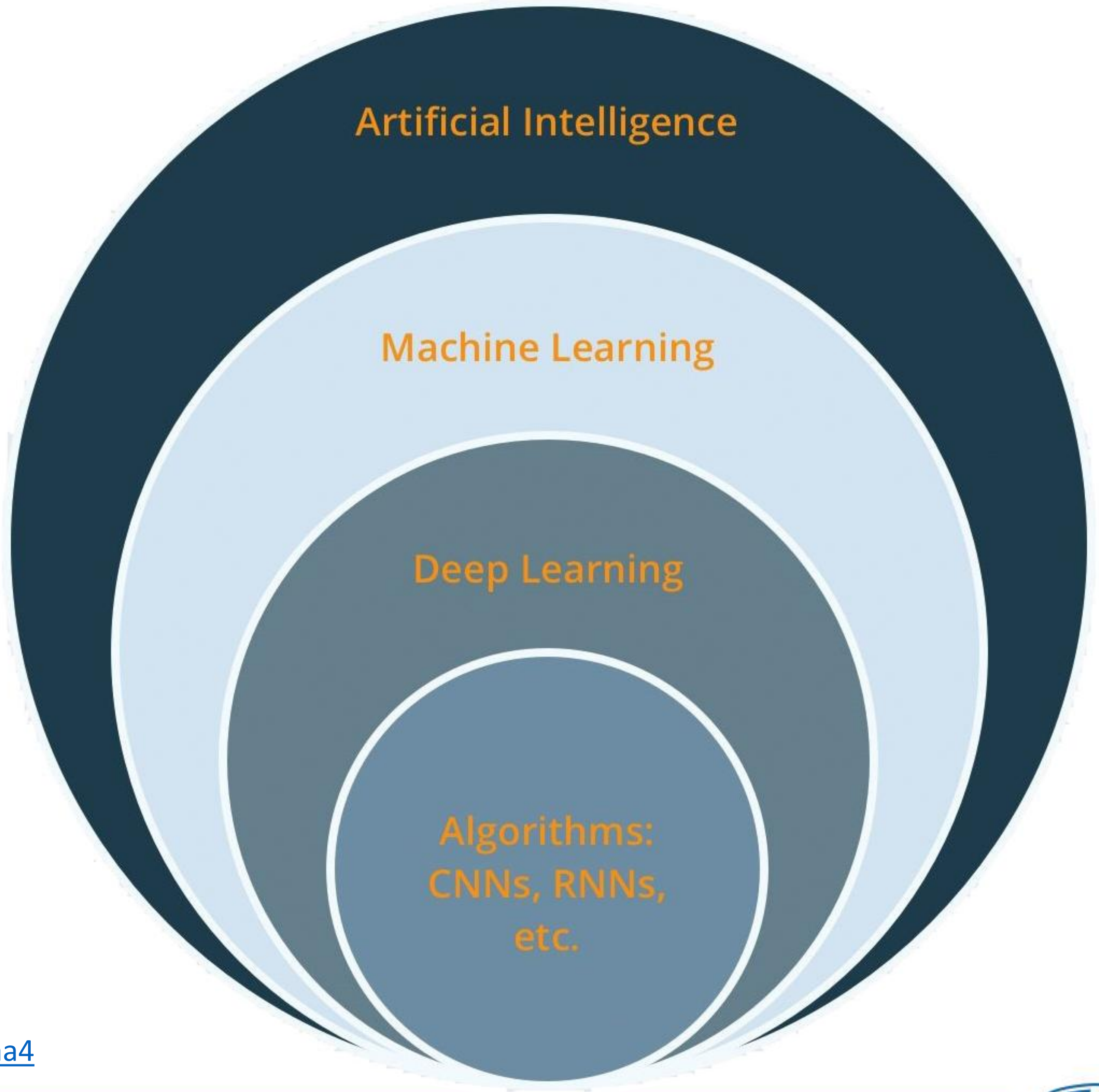
Shaghayegh Khalighiyan

INSTRUCTOR:
Dr. Yasser Alginahi

Oct 27, 2023

University of Windsor

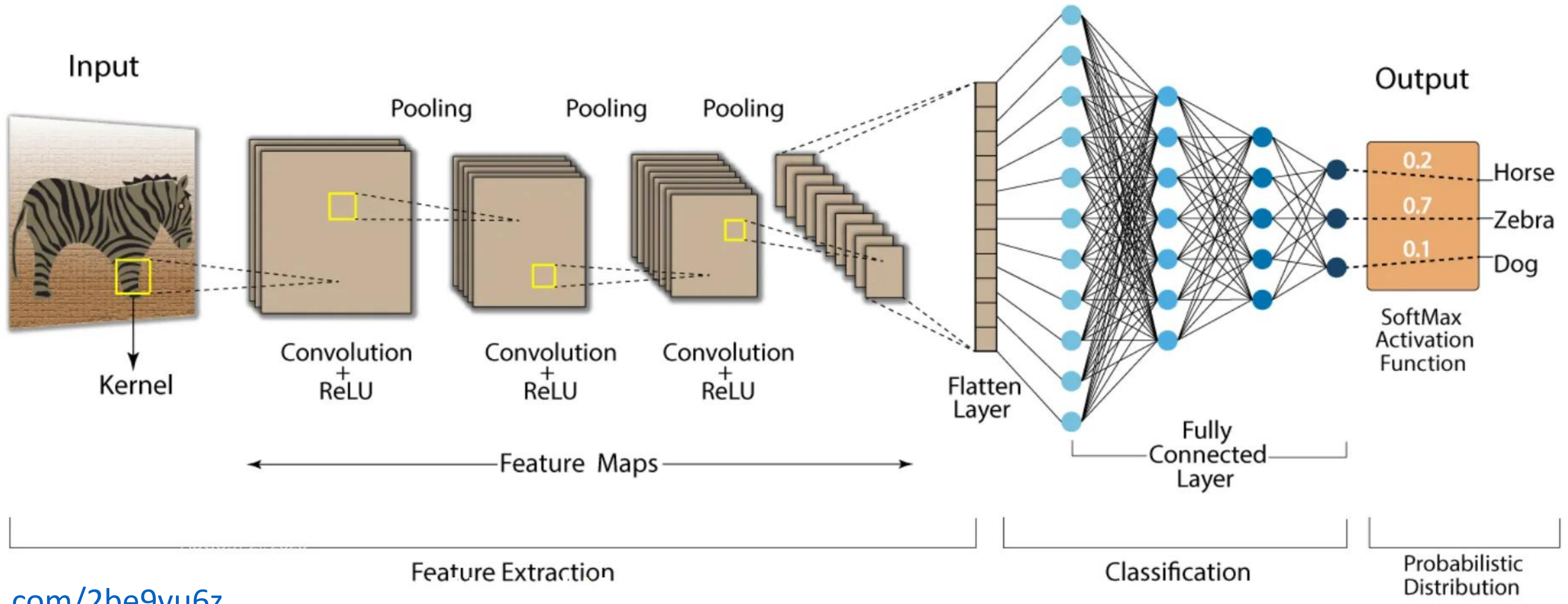
Agenda:



source: https://tinyurl.com/46p33aa4

What is CNN?

CNNs represent a specialized deep learning neural network structure ideally designed for tasks involving image categorization and the identification of objects. In CNN, an initial image input is converted into a feature map, and this map undergoes a series of convolutional and pooling layers to ultimately generate a predicted outcome.



source: https://tinyurl.com/2be9vu6z

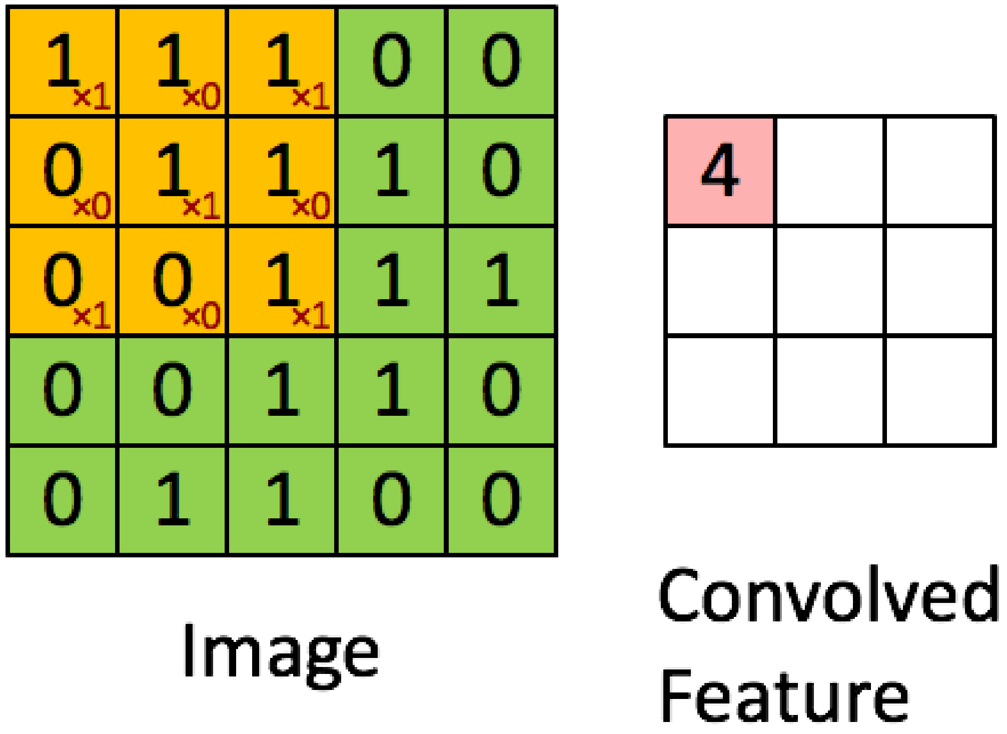
Layers of CNN:

Convolution neural networks have multiple layers that help in extracting information from an image. The four main layers in CNN are:

- Convolution Layer
- Activation Layer (ex. ReLU)
- Pooling Layer
- Fully Connected Layer

Layers of CNN - Convolution Layer:

Convolutional layer extracts features from the input image. It applies a filter or kernel to the input image during a convolution process in order to recognize and extract particular characteristics.

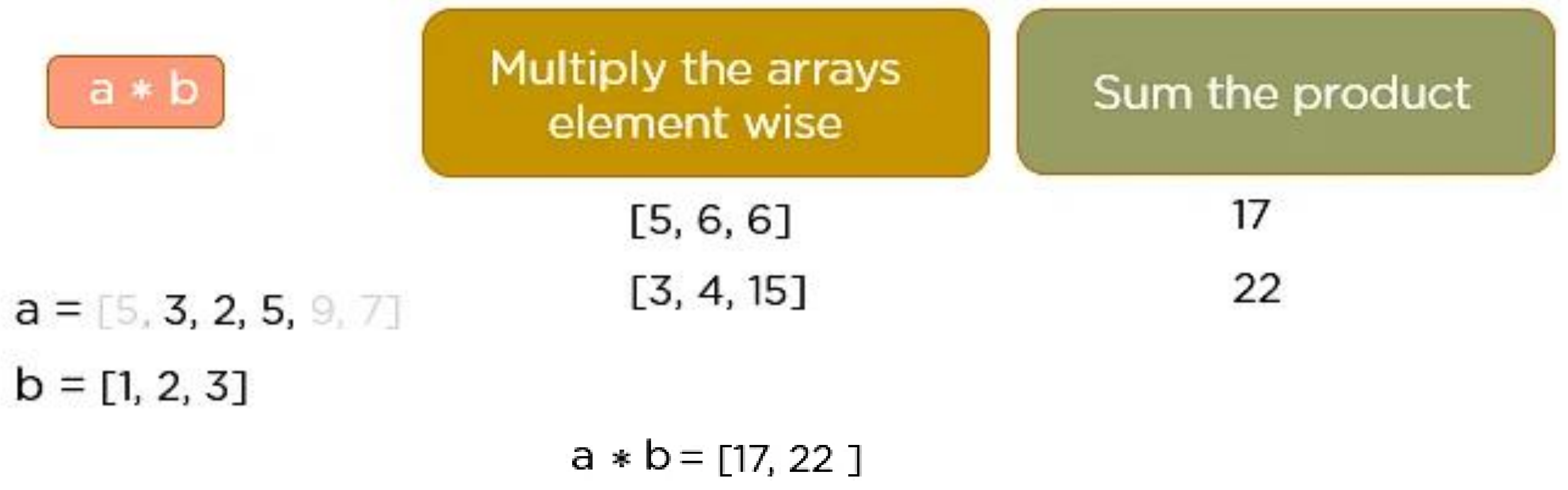


source: https://tinyurl.com/bde3xen4

University of Windsor

Convolution Operation:

The core of any convolutional neural network is the convolution operation. Here you can see this operation through the utilization of two one-dimensional matrices, denoted as 'a' and 'b‘ by applying the element wise multiplication and then summation of the product. This process continues until the convolution operation has been finalized.



source: https://tinyurl.com/yxcbm5st

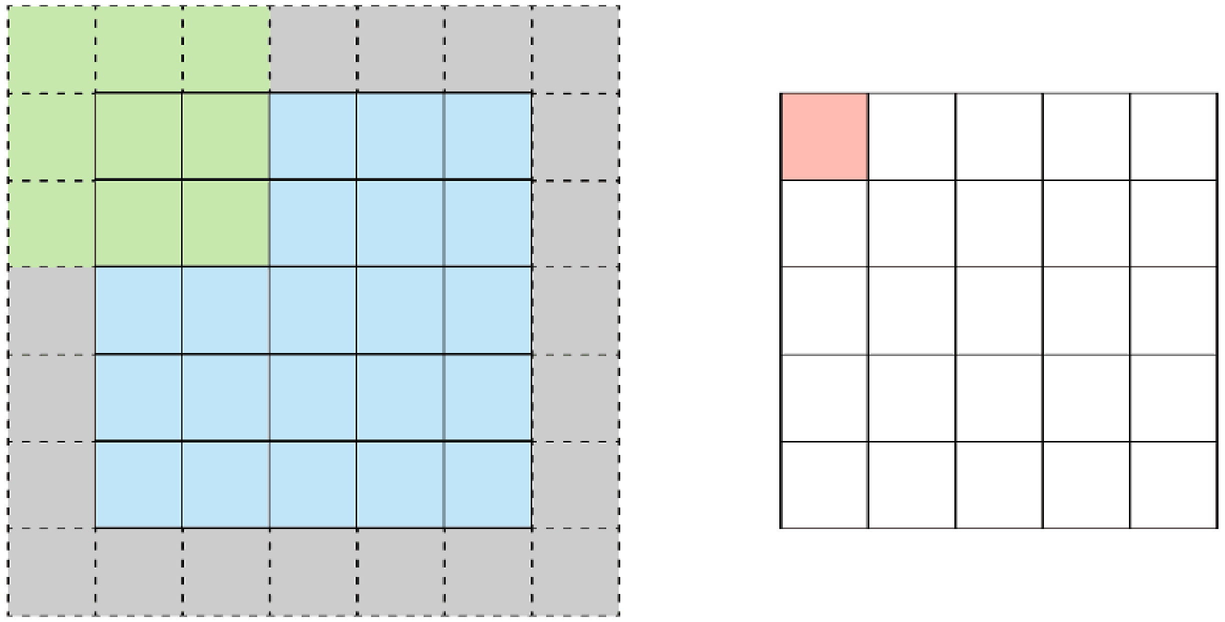
Stride and Padding:

The number of steps we take in each convolutional step is indicated by the stride.

Padding is used to keep the output's dimensions consistent with the input. The process of symmetrically adding zeros to the input matrix is known as padding.

The size of padding can be define by this formula:

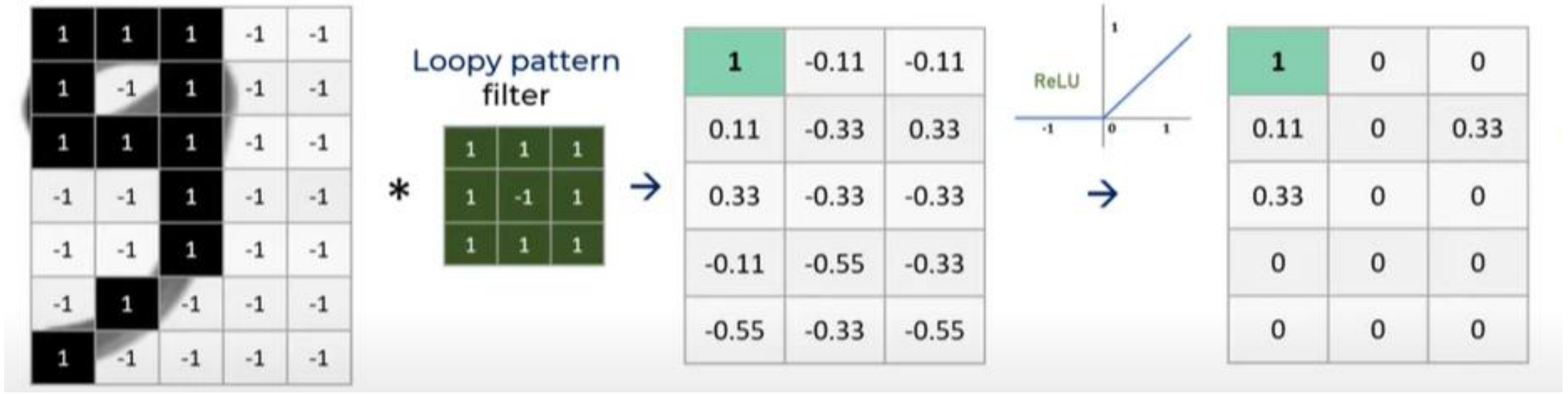
$$(N \times N) * (F \times F) \Rightarrow (N-F+1)\times(N-F+1)$$
$$(N+2p \times N+2p) * (F \times F) \Rightarrow (N+2p-F+1)\times(N+2p-F+1)$$
$$N+2p-F+1 = N$$
$$p = (F-1)/2$$



source: https://tinyurl.com/kkwd6dn7

Layers of CNN-Activation Layer:

The output of the convolution layer will be subjected to an element-wise activation function, such as the ReLU function. In this phase, the output volume is not modified. Activation Layer introduces nonlinearity into the model and speeds up training and computation.

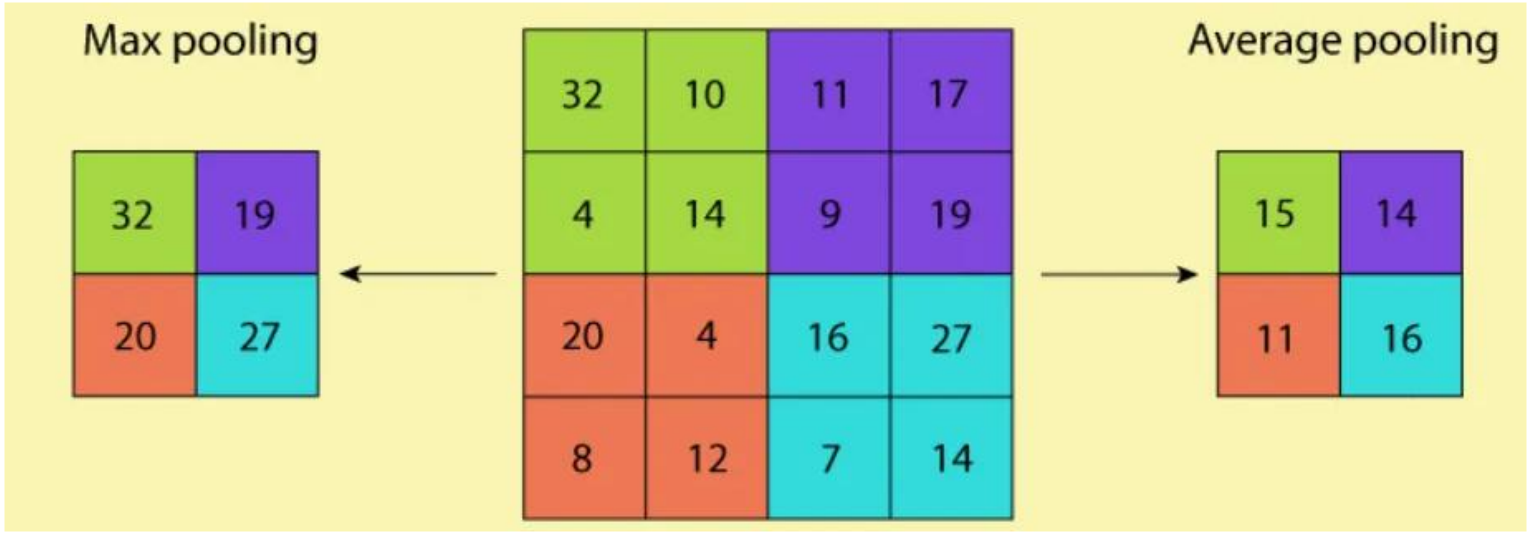


$$\begin{bmatrix} 1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -0.11 & -0.11 \\ 0.11 & -0.33 & 0.33 \\ 0.33 & -0.33 & -0.33 \\ -0.11 & -0.55 & -0.33 \\ -0.55 & -0.33 & -0.55 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0.11 & 0 & 0.33 \\ 0.33 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

source: https://tinyurl.com/5zeh947z

Layers of CNN-Pooling Layer:

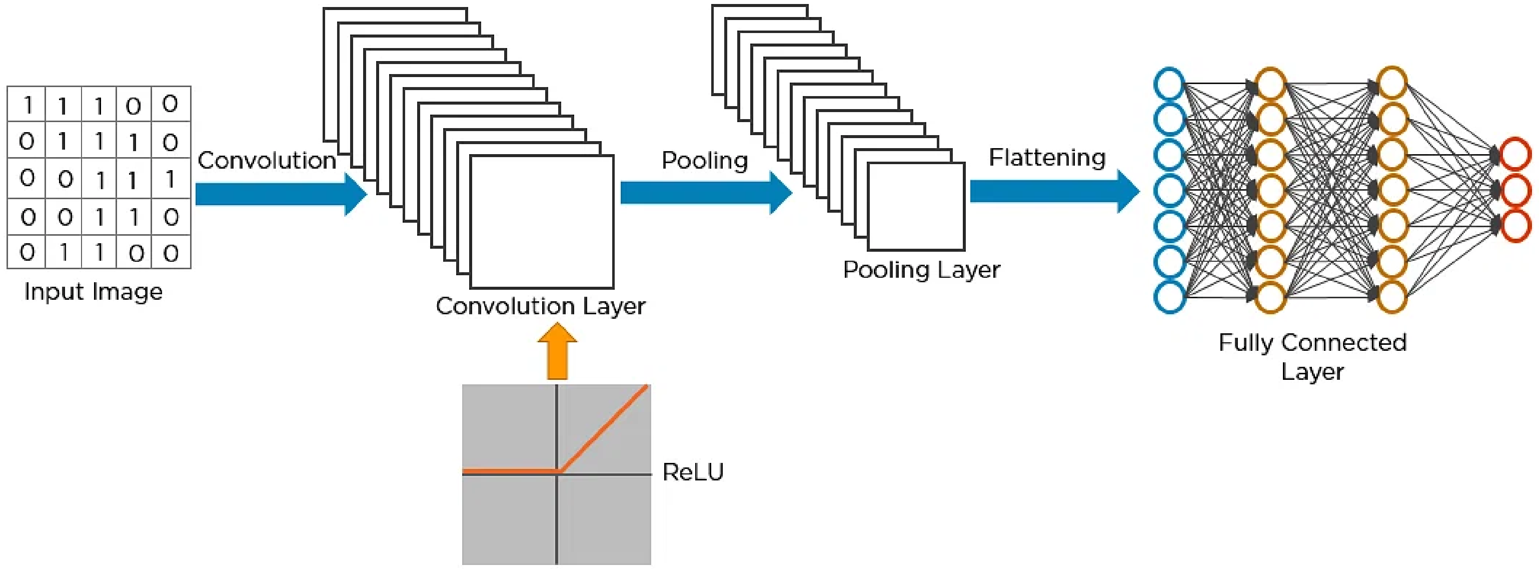
The spatial dimensions of the feature maps generated by the convolutional layer are decreased by the pooling layer. To decrease the size of the feature maps and the computational complexity, it downsamples the data. It also reduces the risk of overfitting. There are two kinds of pooling layer: Max pooling, and Average pooling.



source: https://tinyurl.com/2be9vu6z

Layers of CNN-Fully Connected Layers:

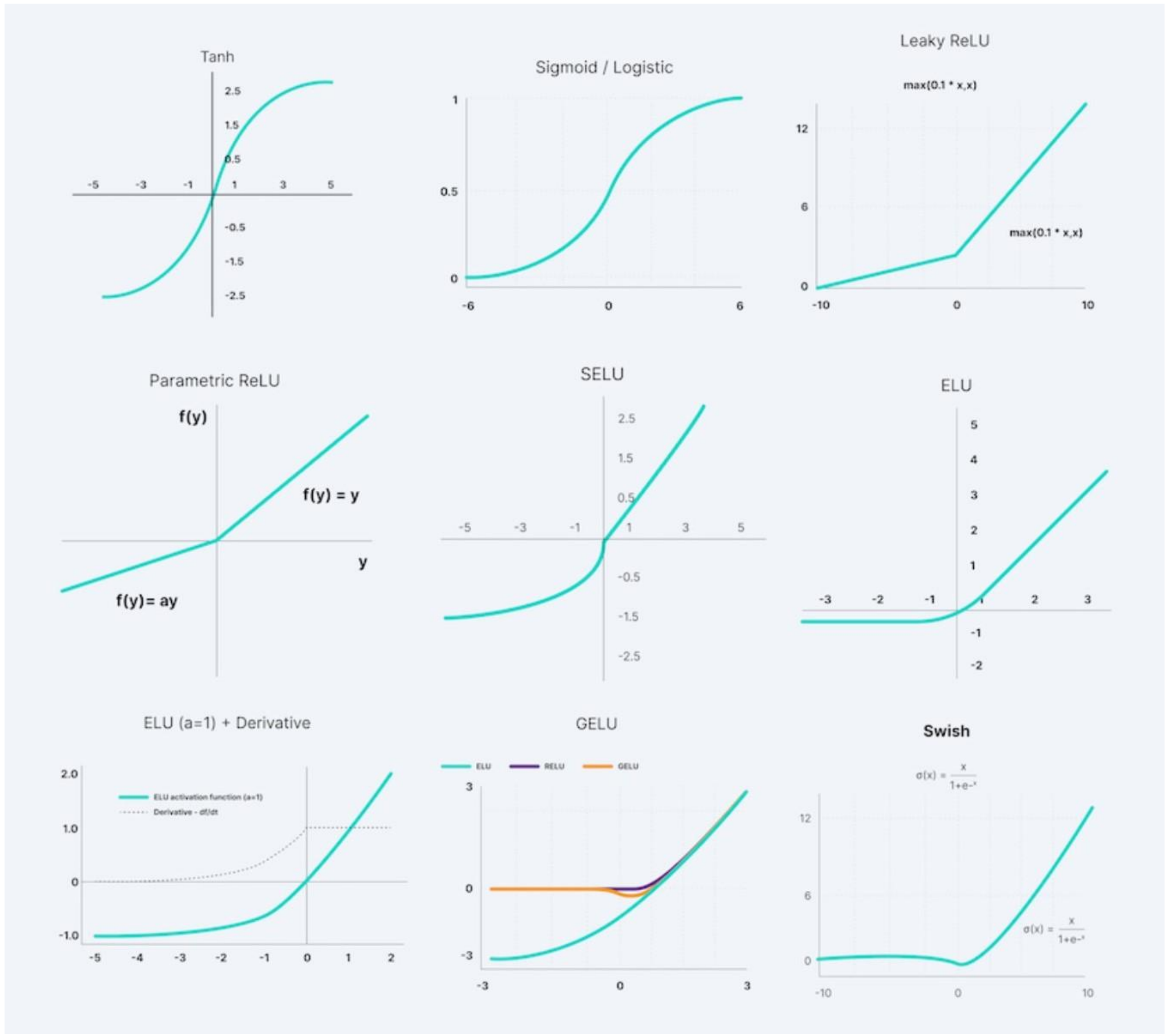
After flattening the features the FC layers uses them to carry out classification tasks. The FC layer usually employs a softmax function or logistic function. Logistic is utilized for binary classification and softmax is for multi-classification.



source: https://tinyurl.com/yxcbm5st

Activation Function:

The decision of whether or not to activate a neuron is made by an activation function. This means that it will use less complex mathematical operations to determine whether or not the neuron's input to the network is significant during the prediction process. The main purpose of using an activation function is to add non-linearity to the neural network.



source: https://tinyurl.com/mp4sphut

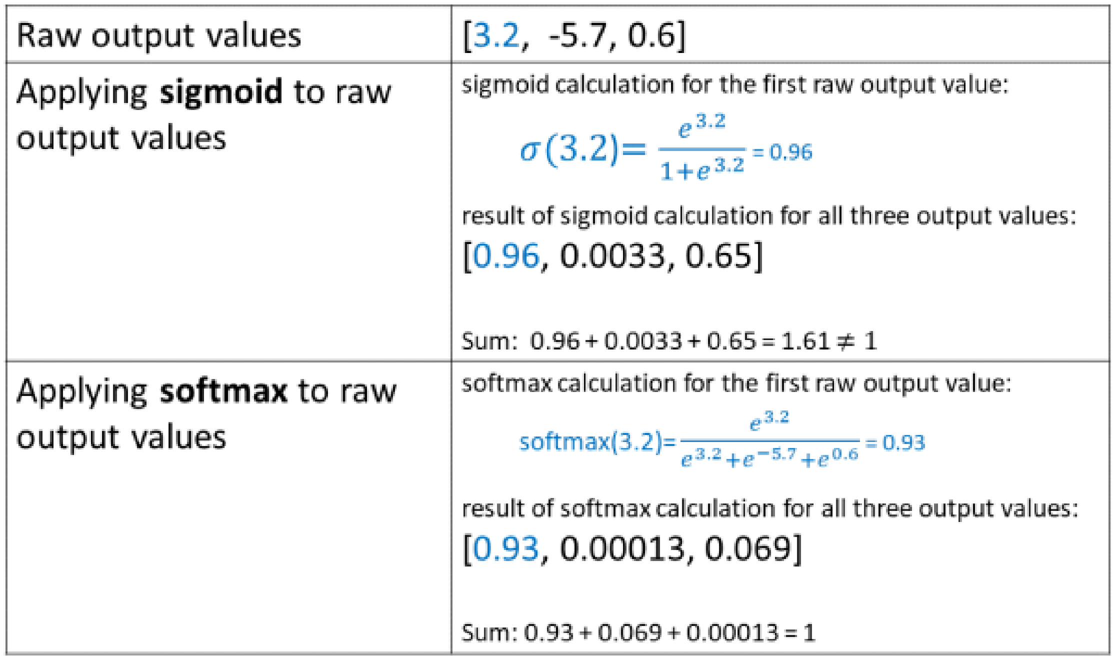
Activation Function:

- **SoftMax:**

  The SoftMax function is a combination of several Sigmoids. The relative probability are computed. The SoftMax function yields the probability of each class, much like the sigmoid/logistic activation function. When it comes to multi-class classification, it is most frequently utilized as an activation function for the neural network's last layer.

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\Sigma_j \exp(z_j)}$$

Sigmoid and SoftMax calculation example:



Raw output values	[3.2, -5.7, 0.6]
Applying **sigmoid** to raw output values	sigmoid calculation for the first raw output value: $\sigma(3.2)= \frac{e^{3.2}}{1+e^{3.2}} = 0.96$ result of sigmoid calculation for all three output values: [0.96, 0.0033, 0.65] Sum: $0.96 + 0.0033 + 0.65 = 1.61 \neq 1$
Applying **softmax** to raw output values	softmax calculation for the first raw output value: $\text{softmax}(3.2)= \frac{e^{3.2}}{e^{3.2}+e^{-5.7}+e^{0.6}} = 0.93$ result of softmax calculation for all three output values: [0.93, 0.00013, 0.069] Sum: $0.93 + 0.069 + 0.00013 = 1$

source: https://tinyurl.com/bddne9zk

Loss Function:

A loss function evaluates how well a neural network predicts the training set of data by comparing the target and predicted output values.

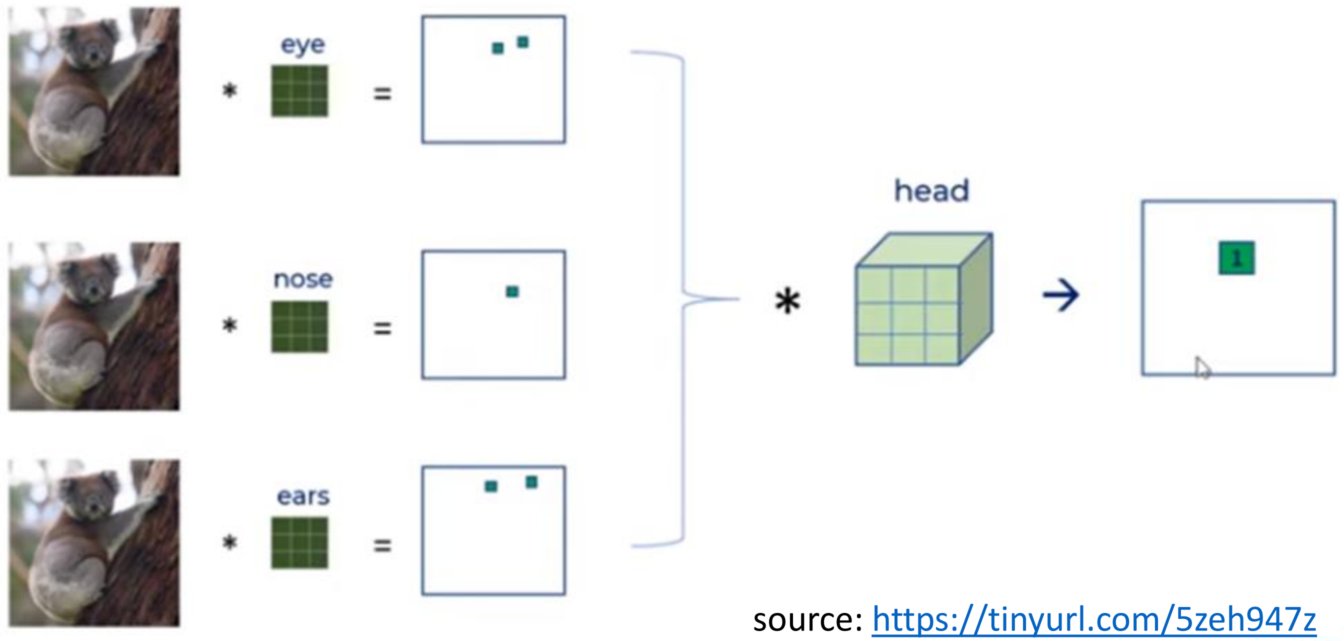
In supervised learning, there exist two primary categories of loss functions, which align with the two major classes of neural networks: regression and classification:

- Regression Loss Functions → Ex. Mean Squared Error, Mean Absolute Error
- Classification Loss Functions → Ex. Binary Cross-Entropy, Categorical Cross-Entropy

Feature Extraction Example:

Let’s consider a CNN model that detects koala.

Three filters are applied to detect eye, nose and ears, and another convolutional operation can be applied again to detect head. We are aggregating the result using a different filter for head and pixel 1 shows koala’s head.

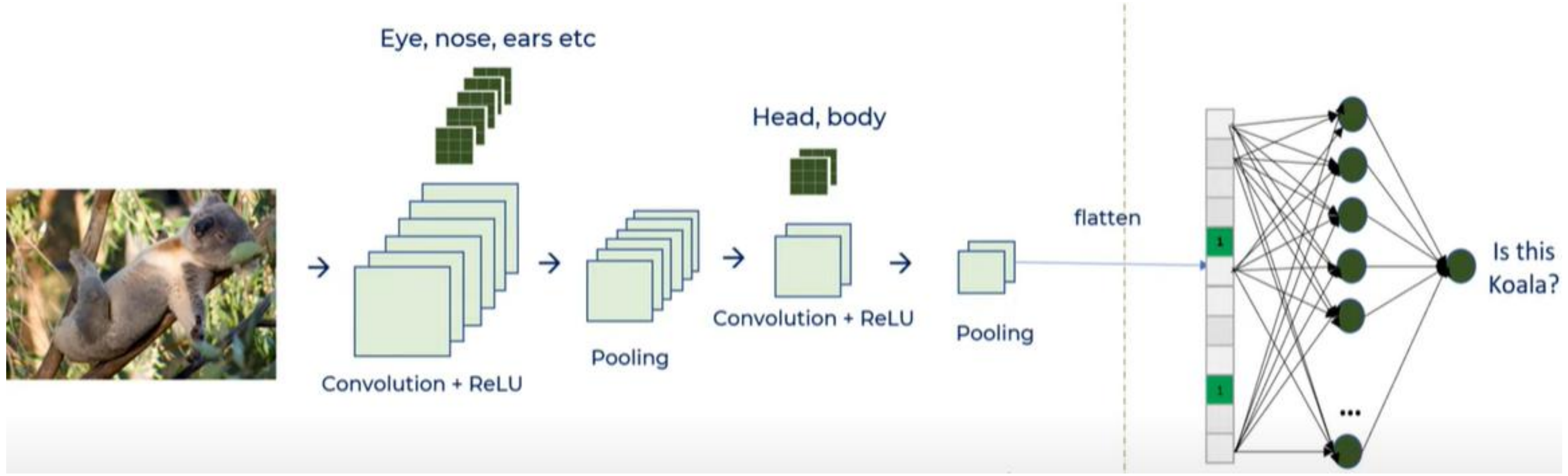


source: https://tinyurl.com/5zeh947z

Feature Extraction Example:

ReLU operation will be performed to brings nonlinearity to our model by changing the negative numbers to zero and the positive numbers will be kept as it is.

The issue of too much computation is not addressed yet. We are getting the same size of image. Here, the pooling layer is used to reduce the dimension.



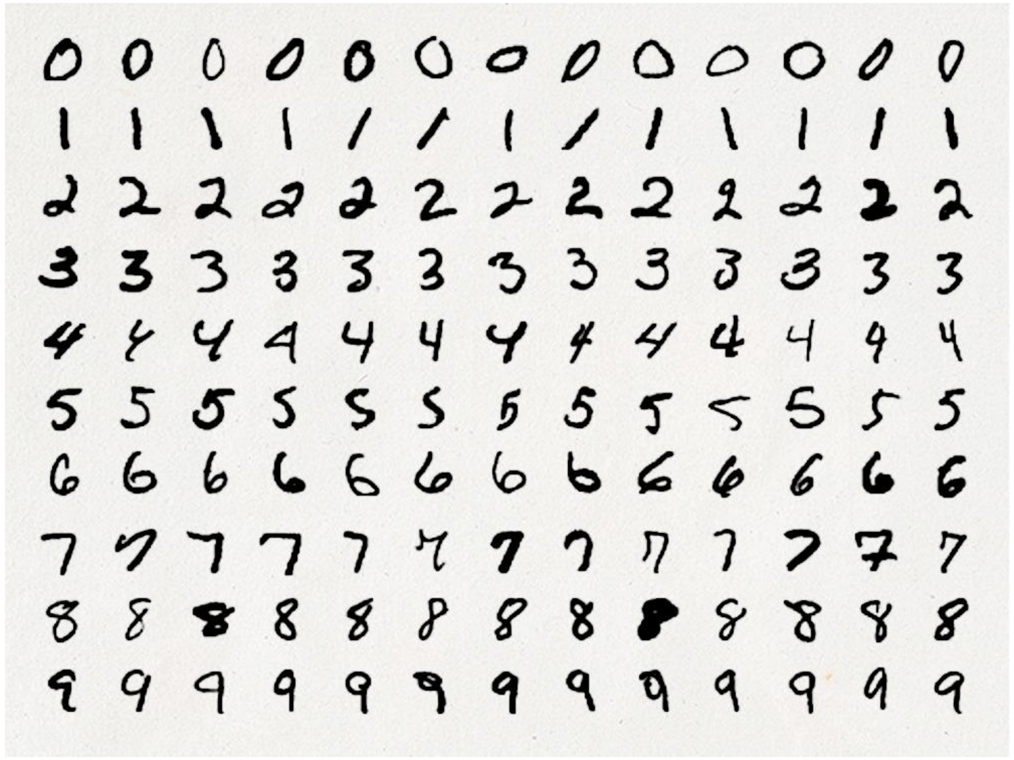
source: https://tinyurl.com/5zeh947z

Example-Detection of Handwritten Digits:

Dataset:

MNIST is a collection of 60,000 tiny square grayscale photographs, with the size of 28*28, comprising handwritten single digits between 0 and 9.

The goal is to categorize a provided image of a handwritten digit into one of ten classes, which together represent the integer values 0 through 9. The MNIST dataset is loaded using Keras in this example.



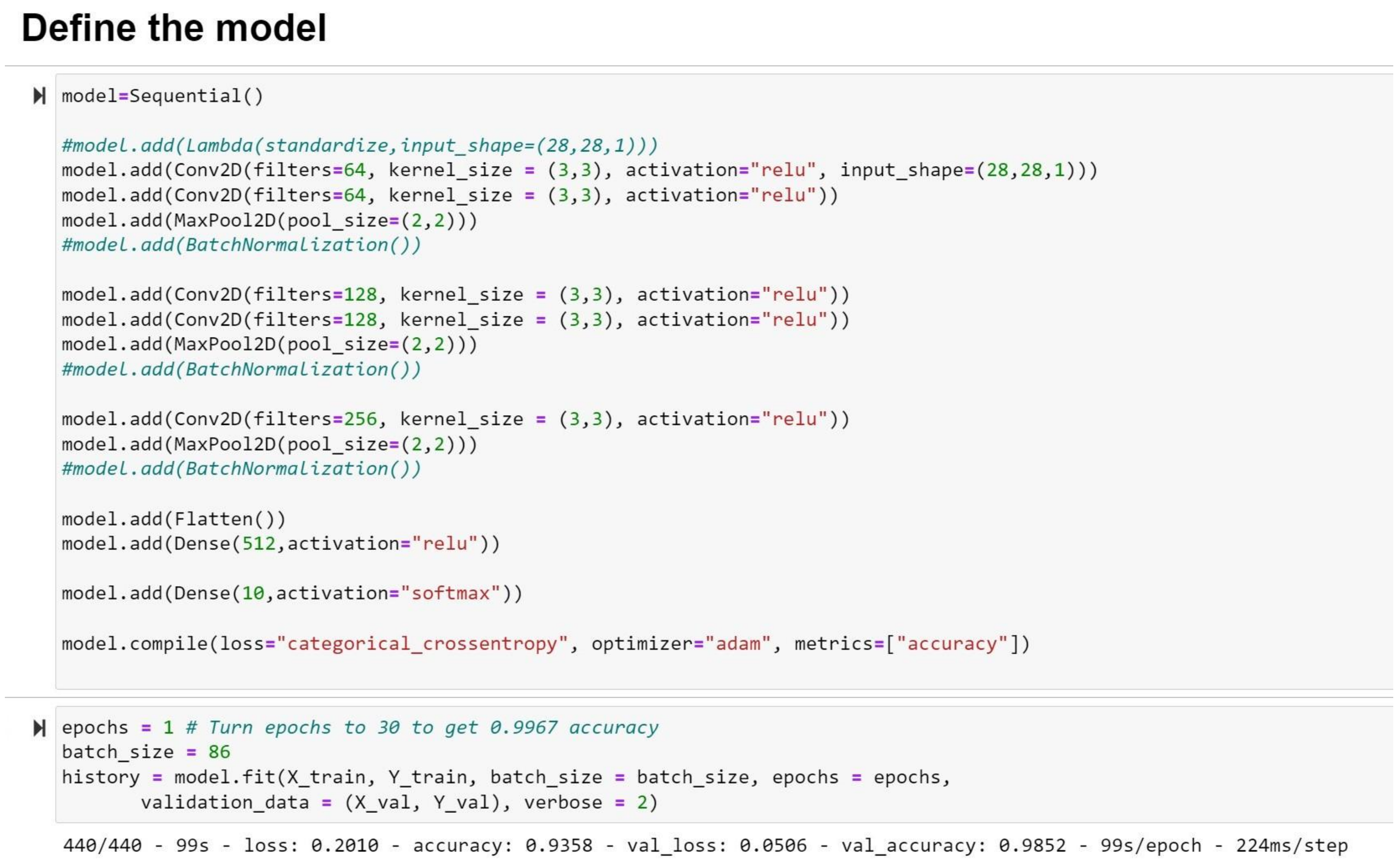
source: https://tinyurl.com/bdhxzca6

Example-Detection of Handwritten Digits:

- For the first two layers 64 filters and 128 filters for the two second layers and 256 for the last one have been chosen.
- The second layer is the pooling (MaxPool2D) layer. It looks at the 2 neighboring pixels and picks the maximal value.
- The Flatten layer is use to convert the final feature maps into a one single 1D vector.
- In the end two fully-connected (Dense) layers which are artificial neural network (ANN) classifiers are added. The final layer (Dense(10,activation="softmax")) yields the probability distribution of each class's net output.

Code:

Code Link Access:
https://colab.research.google.com/drive/1lgJXWeZnD03-hw9PPJGh2_slPfaqnTmU?usp=sharing



Define the model

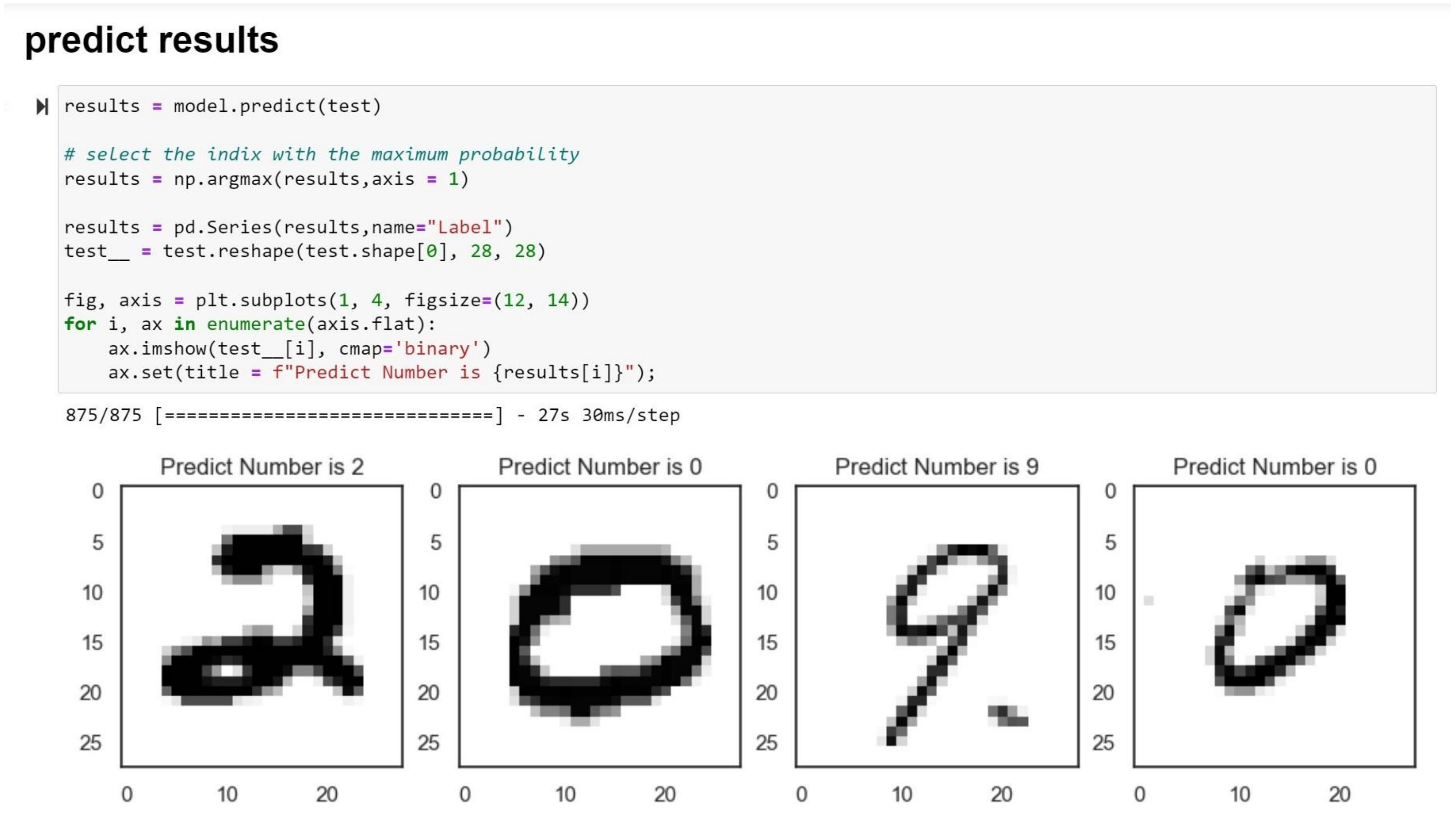
```
model=Sequential()

#model.add(Lambda(standardize,input_shape=(28,28,1)))
model.add(Conv2D(filters=64, kernel_size = (3,3), activation="relu", input_shape=(28,28,1)))
model.add(Conv2D(filters=64, kernel_size = (3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))
#model.add(BatchNormalization())

model.add(Conv2D(filters=128, kernel_size = (3,3), activation="relu"))
model.add(Conv2D(filters=128, kernel_size = (3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))
#model.add(BatchNormalization())

model.add(Conv2D(filters=256, kernel_size = (3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))
#model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(512,activation="relu"))

model.add(Dense(10,activation="softmax"))

model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

```
epochs = 1 # Turn epochs to 30 to get 0.9967 accuracy
batch_size = 86
history = model.fit(X_train, Y_train, batch_size = batch_size, epochs = epochs,
       validation_data = (X_val, Y_val), verbose = 2)
```

```
440/440 - 99s - loss: 0.2010 - accuracy: 0.9358 - val_loss: 0.0506 - val_accuracy: 0.9852 - 99s/epoch - 224ms/step
```

source: https://tinyurl.com/3w3kxmjc

Code:



predict results

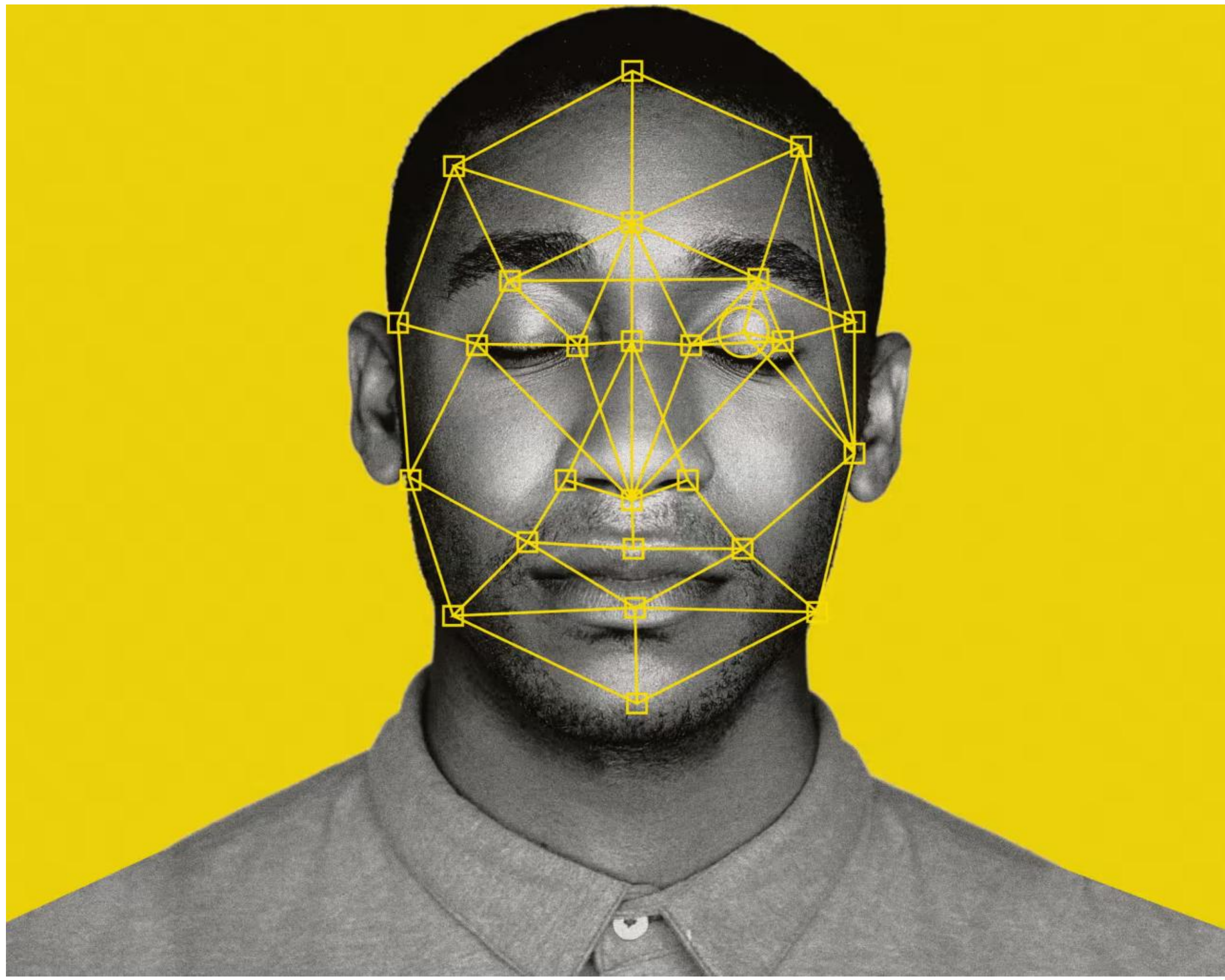
```python
results = model.predict(test)

# select the indix with the maximum probability
results = np.argmax(results,axis = 1)

results = pd.Series(results,name="Label")
test__ = test.reshape(test.shape[0], 28, 28)

fig, axis = plt.subplots(1, 4, figsize=(12, 14))
for i, ax in enumerate(axis.flat):
    ax.imshow(test__[i], cmap='binary')
    ax.set(title = f"Predict Number is {results[i]}");
```

```
875/875 [==============================] - 27s 30ms/step
```

source: https://tinyurl.com/3w3kxmjc

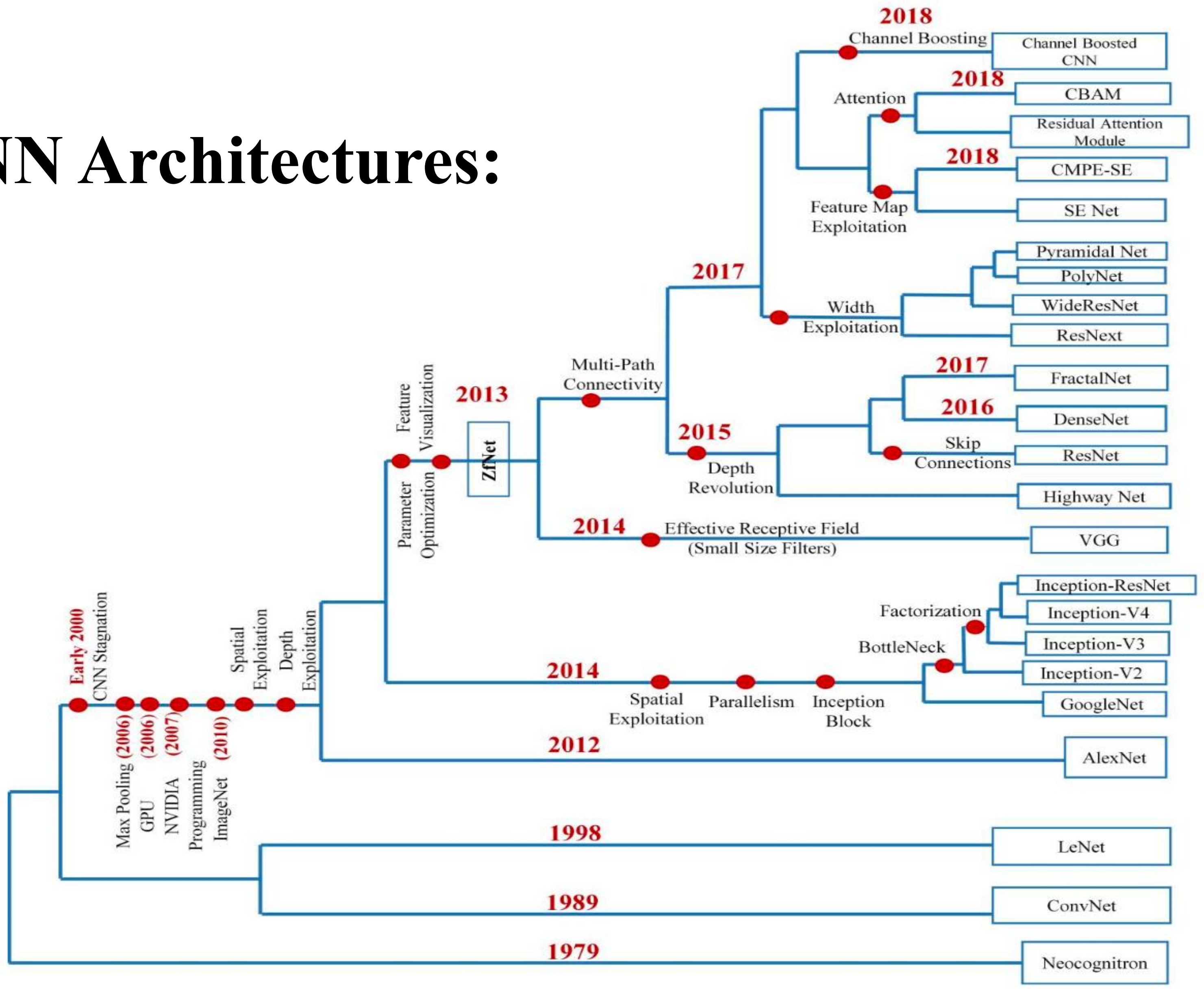
CNN Applications:

- Image Classification
- Recommender Systems
- Image Retrieval
- Face Recognition
- Optical Character Recognition (OCR)



source: https://tinyurl.com/56x35mv8

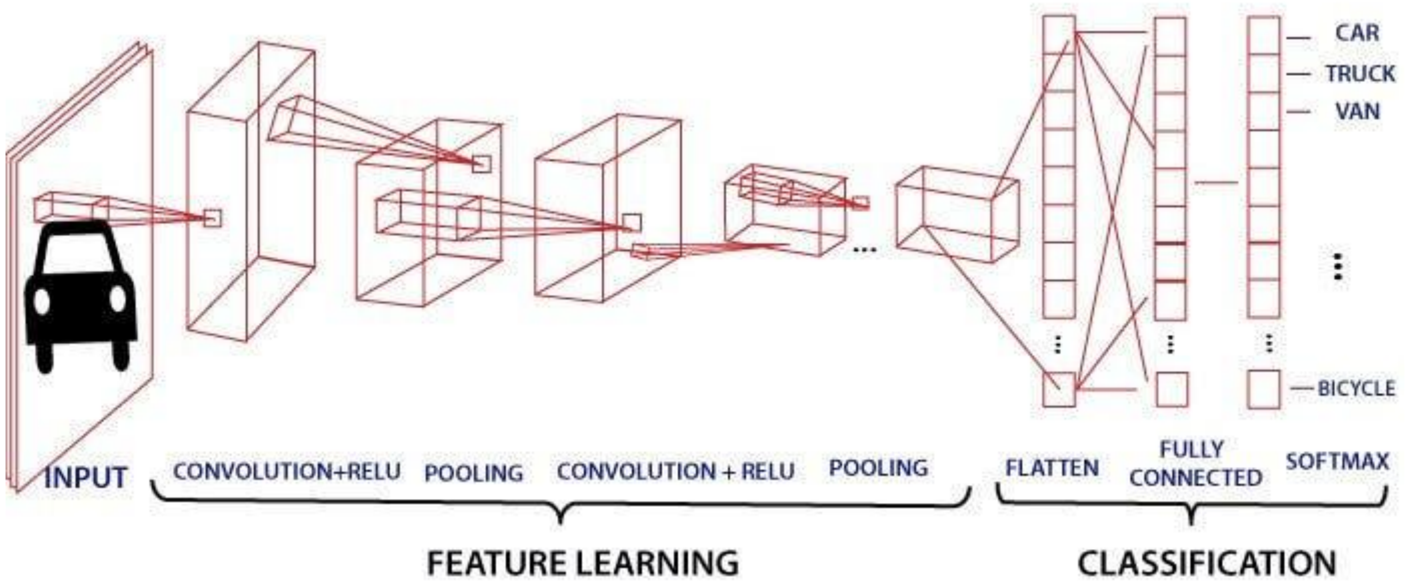
History of CNN Architectures:



source: https://tinyurl.com/2uxs8xz9

Benefits:

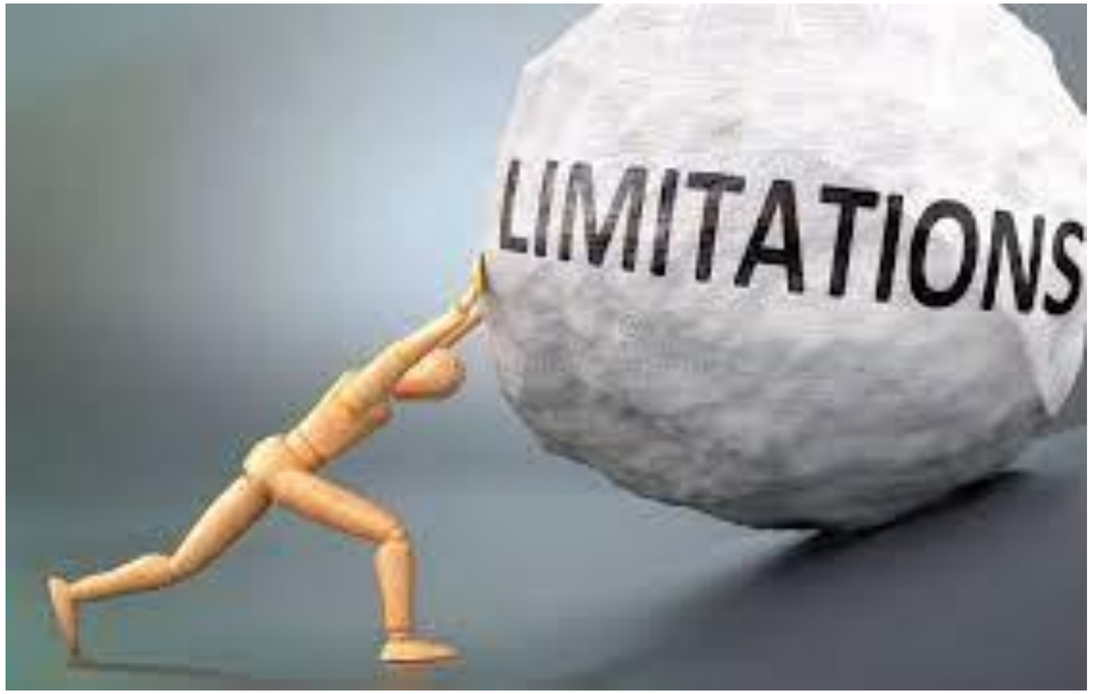
- Feature extraction
- Spatial invariance
- Robust to noise
- Transfer learning
- Performance



source: https://tinyurl.com/4bhtkywf

Limitations:

- Computational cost
- Overfitting
- Lack of interpretability
- Limited to grid-like structures



source: https://tinyurl.com/3y6hyrnw

Conclusion:

Convolutional Neural Networks (CNNs) are a strong deep learning architecture that are excellent for tasks involving object and image recognition. CNNs have shown cutting-edge performance on a variety of computer vision tasks thanks to their capacity to automatically extract relevant features, handle noisy images, and utilize pre-trained models.

They do, however, have certain drawbacks, such as a high computational cost, overfitting, interpretability issues, and a restricted capacity to handle irregular shapes, CNNs are still a popular option for a variety of computer vision tasks, and in the years to come, they'll probably remain a major focus of research and development.

- https://www.youtube.com/watch?v=vT1JzLTH4G4
- https://www.youtube.com/watch?v=zfiSAzpy9NM
- https://www.youtube.com/watch?v=Jy9-aGMB_TE&t=589s
- https://www.youtube.com/watch?v=HGwBXDKFk9I

References:

[1] “Convolutional neural network: Benefits, types, and applications,” Datagen, https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/ (accessed Oct. 30, 2023).

[2] dshahid380, “Convolutional Neural Network,” Medium, https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529 (accessed Oct. 30, 2023).

[3] N. Shahriar, “What is Convolutional Neural Network - CNN (deep learning),” Medium, https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5 (accessed Oct. 30, 2023).

[4] A. Biswal, “Convolutional Neural Network tutorial [update],” Simplilearn.com, https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network (accessed Oct. 30, 2023).

[5] P. Baheti, “Activation functions in neural networks [12 types &amp; use cases],” V7, https://www.v7labs.com/blog/neural-networks-activation-functions (accessed Oct. 30, 2023).