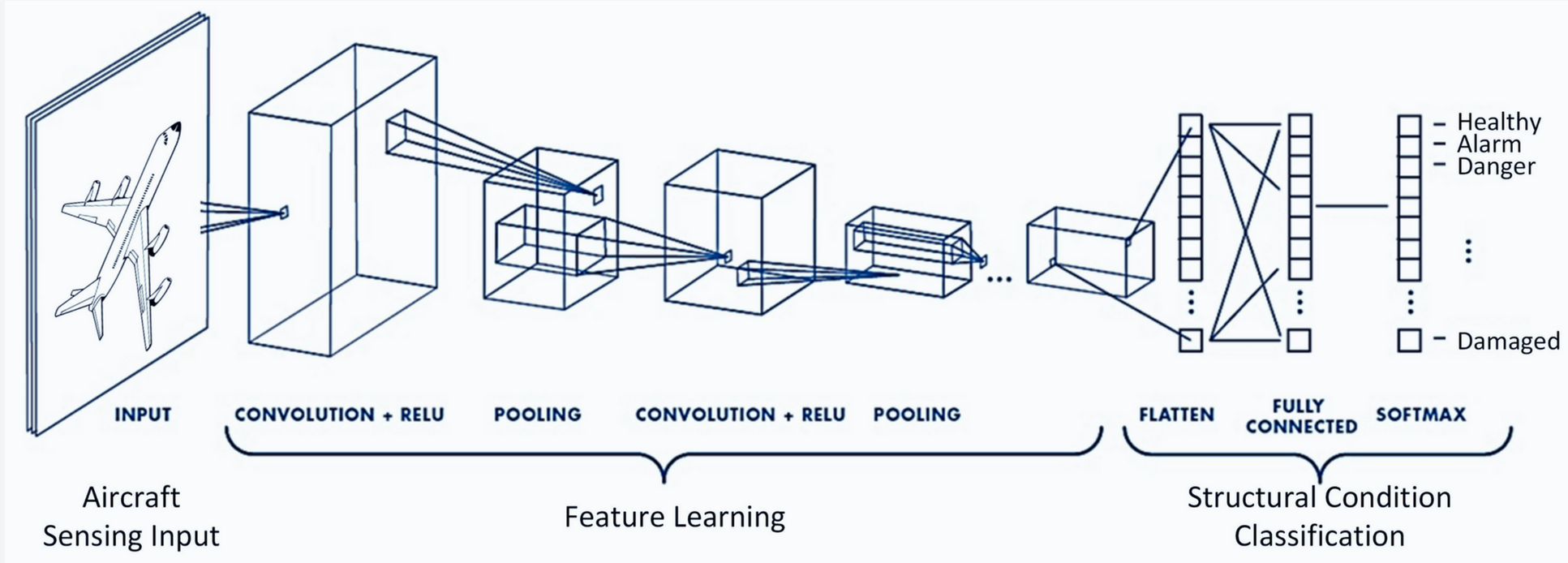
TECH ACADEMY

# Handwritten English Alphabet Recognition with Convolutional Neural Network



TECH ACADEMY

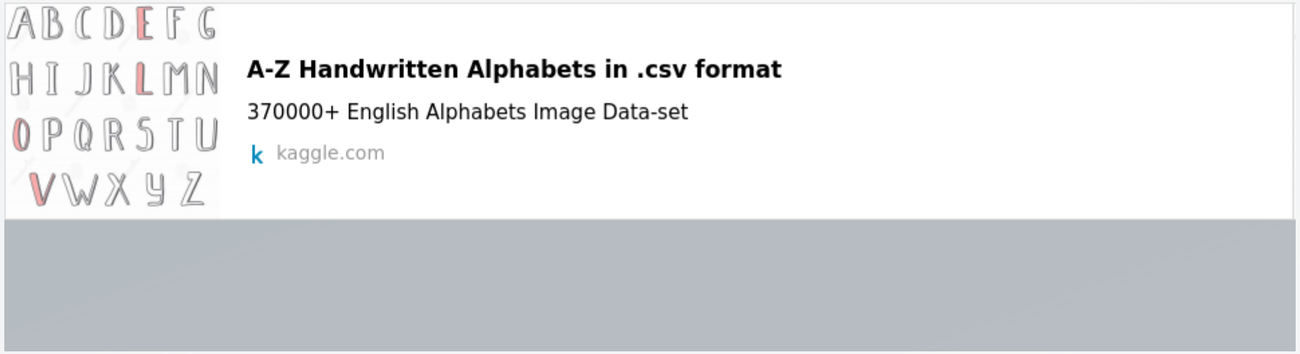
# CONTENT

TECH ACADEMY

# İNTRODUCTİON

Convolutional neural network (CNN) is a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization. Vanishing gradients and exploding gradients, seen during backpropagation in earlier neural networks, are prevented by using regularized weights over fewer connections. For example, for each neuron in the fully-connected layer 10,000 weights would be required for processing an image sized 100 × 100 pixels. However, applying cascaded convolution (or cross-correlation) kernels, only 25 neurons are required to process 5x5-sized tiles.

# DATA SOURCE, DATA CLEANİNG




The dataset contains 372 450 handwritten images in size 28 x 28 pixels, each alphabet in the image is centre fitted to 20 x 20 pixel box.

Data cleaning (sometimes also known as data cleansing or data wrangling) is an important early step in the data analytics process. This crucial exercise, which involves preparing and validating data, usually takes place before your core analysis. Data cleaning is not just a case of removing erroneous data, although that's often part of it. The majority of work goes into detecting **rogue data** and (wherever possible) correcting it.

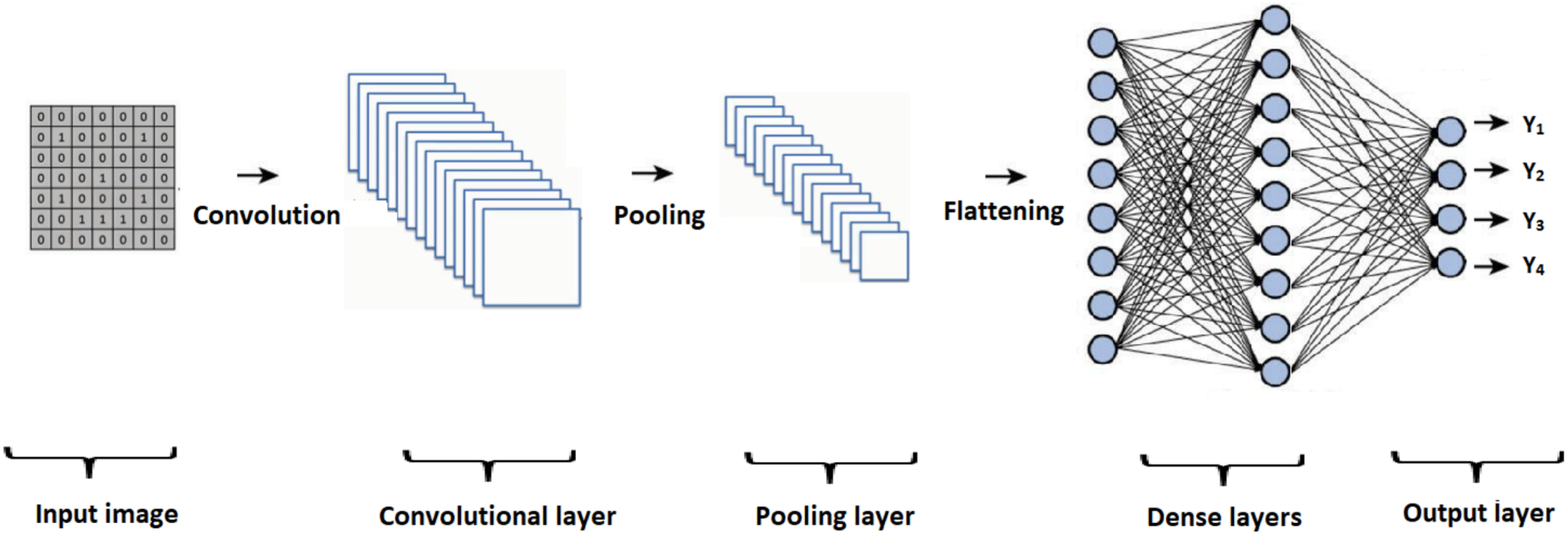
'**Rogue data'** includes things like incomplete, **inaccurate, irrelevant, corrupt or incorrectly formatted** data. The process also involves **deduplicating**, or '**deduping'**. This effectively means merging or removing identical data points.

```
print("Count of Images Before deduplicating process",len(df))
df.drop_duplicates(inplace=True)
print("Count of Images After deduplicating process",len(df))
```

```
Count of Images Before deduplicating process 372450
Count of Images After deduplicating process 201095
```

# Feature Extraction

- Convolutional layer
- Max Pooling



# Convolutional layer

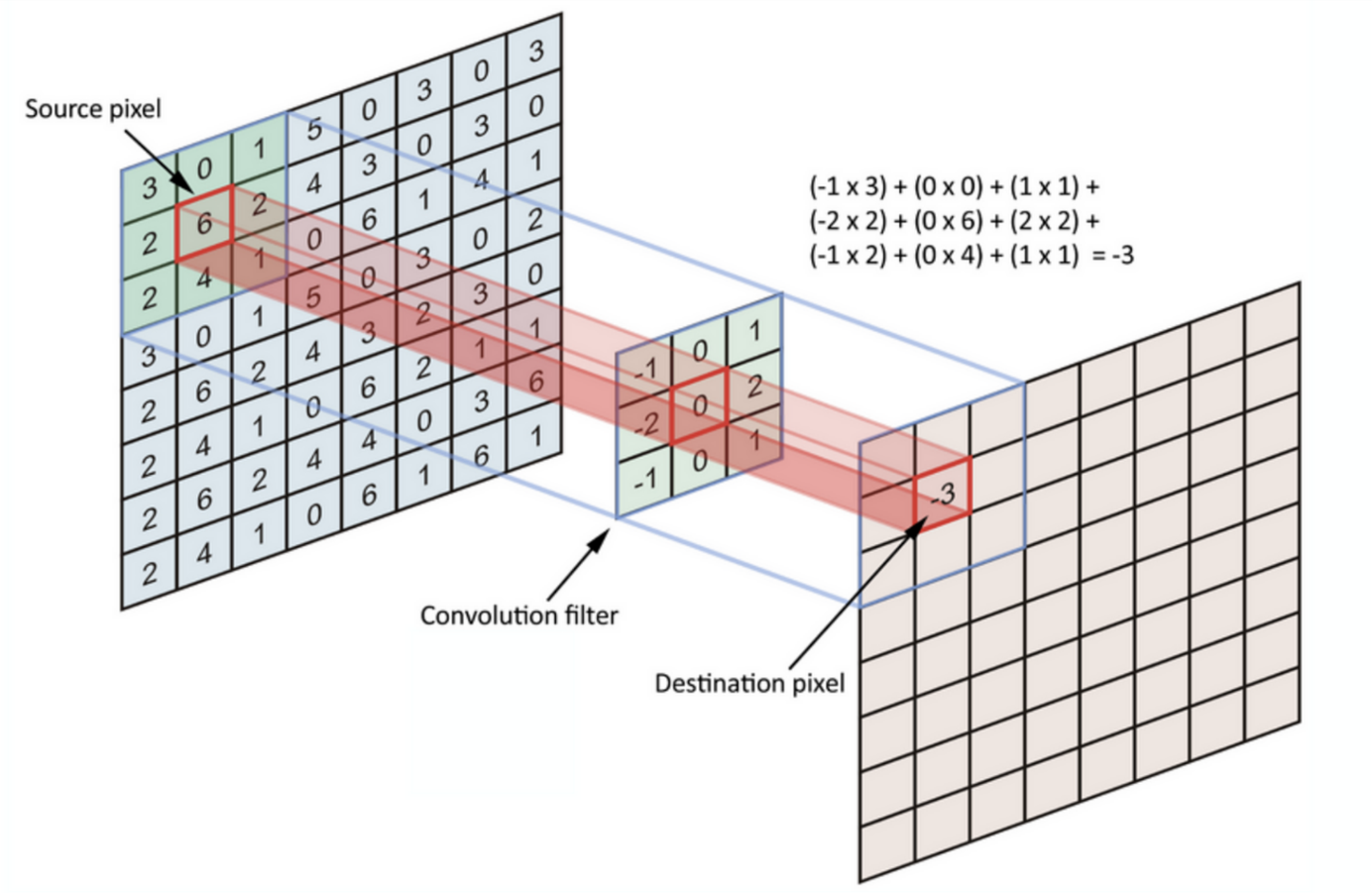
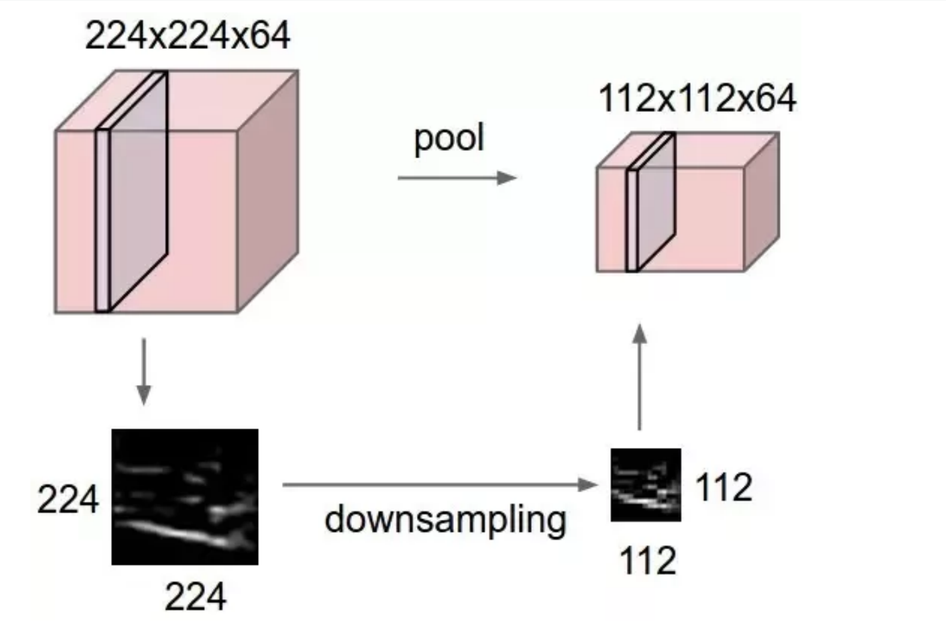
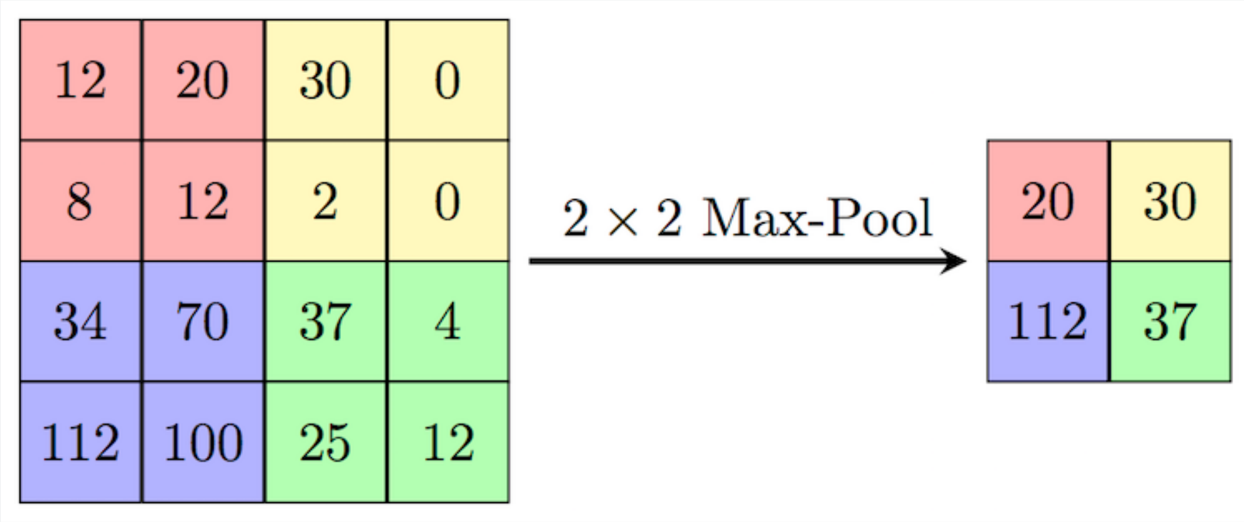



Image-based soft sensors are of interest in process industries due to their cost-effective and non-intrusive properties. Unlike most multivariate inputs, images are highly dimensional, requiring the use of feature extractors to produce lower dimension representations. These extractors have a large impact on final sensor performance. Traditional texture feature extraction methods consider limited feature types, requiring expert knowledge to select and may be sensitive to changing imaging conditions. Deep learning methods are an alternative which does not suffer these drawbacks. A specific deep learning method, Convolutional Neural Networks (CNNs), mitigates the curse of dimensionality inherent in fully connected networks but must be trained, unlike other feature extractors.

# Max Pooling



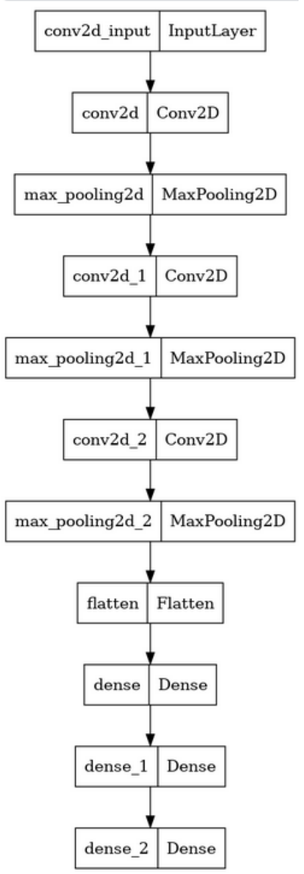

A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input image will result in a different feature map. This can happen with re-cropping, rotation, shifting, and other minor changes to the input image.

A common approach to addressing this problem from signal processing is called down sampling. This is where a lower resolution version of an input signal is created that still contains the large or important structural elements, without the fine detail that may not be as useful to the task.

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned

# Modeling

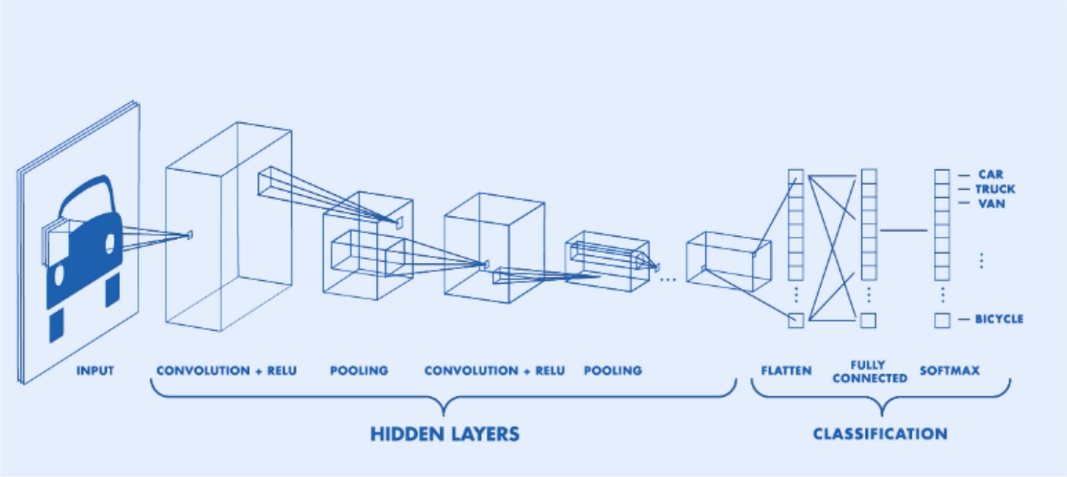
```
model=Sequential()
model.add(Conv2D(16,(3,3),activation="relu"))
model.add(MaxPooling2D(strides=(1,1)))
model.add(Conv2D(8,(3,3),activation="relu"))
model.add(MaxPooling2D(strides=(1,1)))
model.add(Conv2D(8,(3,3),activation="relu"))
model.add(MaxPooling2D(strides=(1,1)))
model.add(Flatten())
model.add(Dense(128,activation="relu"))
model.add(Dense(64,activation="relu"))
model.add(Dense(26,activation="softmax"))
model.compile(loss=losses.categorical_crossentropy,
              metrics=["accuracy"],optimizer=SGD(learning_rate=0.05))

checkpoint = ModelCheckpoint(filepath="model_sgd.pkl",monitor="val_accuracy",
                             verbose=2,save_best_only=True,mode="max")

model.fit(datagen.flow(X_train, y_train, batch_size=128,subset="training"),
          validation_data=datagen.flow(X_train, y_train, batch_size=128,subset="validation"),
          epochs=1,batch_size=128,verbose=2,callbacks=[checkpoint])
```

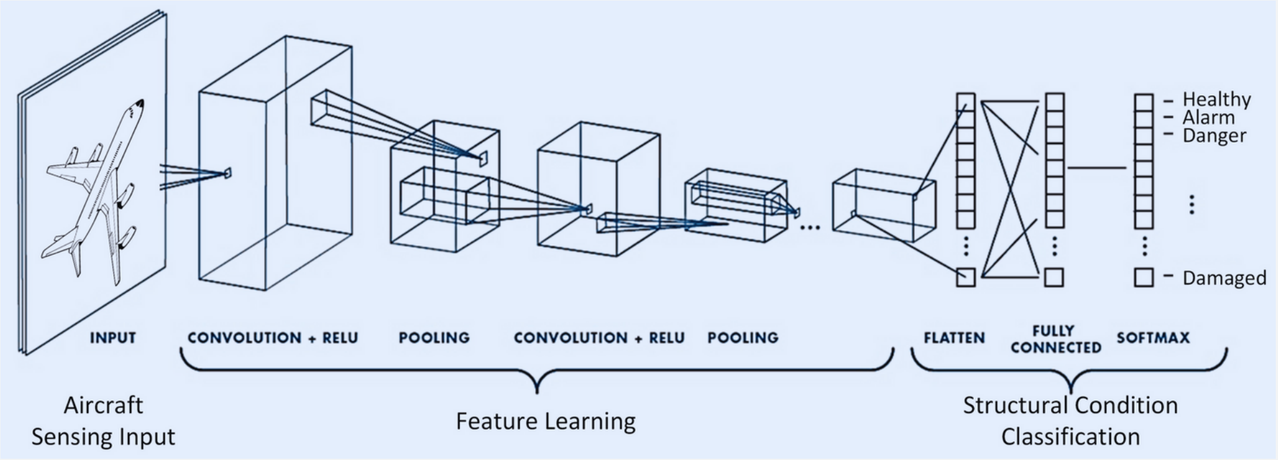



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, None, None, 16)    160

 max_pooling2d (MaxPooling2D  (None, None, None, 16)   0
 )

 conv2d_1 (Conv2D)           (None, None, None, 8)     1160

 max_pooling2d_1 (MaxPooling  (None, None, None, 8)    0
 2D)

 conv2d_2 (Conv2D)           (None, None, None, 8)     584

 max_pooling2d_2 (MaxPooling  (None, None, None, 8)    0
 2D)

 flatten (Flatten)           (None, None)              0

 dense (Dense)               (None, 128)               369792

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 26)                1690

=================================================================
Total params: 381,642
Trainable params: 381,642
Non-trainable params: 0
_________________________________________________________________
```

# Why this model/architecture?

- Convolutional neural networks are multi-layer neural networks that are really good at getting the features out of data. They work well with images and they don't need a lot of pre-processing.
- Using convolutions and pooling to reduce an image to its basic features, you can identify images correctly.
- It's easier to train CNN models with fewer initial parameters than with other kinds of neural networks. You won't need a huge number of hidden layers because the convolutions will be able to handle a lot of the hidden layer discovery for you.



# Results

```
print(classification_report(y_predict,y_test))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 1090 |
| 1 | 0.98 | 0.99 | 0.99 | 710 |
| 2 | 0.99 | 0.98 | 0.99 | 1819 |
| 3 | 0.95 | 0.97 | 0.96 | 813 |
| 4 | 0.98 | 0.99 | 0.98 | 926 |
| 5 | 0.97 | 1.00 | 0.99 | 182 |
| 6 | 0.96 | 0.96 | 0.96 | 527 |
| 7 | 0.99 | 0.91 | 0.95 | 639 |
| 8 | 0.97 | 0.99 | 0.98 | 160 |
| 9 | 0.98 | 0.96 | 0.97 | 714 |
| 10 | 0.96 | 0.98 | 0.97 | 488 |
| 11 | 0.99 | 0.98 | 0.99 | 901 |
| 12 | 0.99 | 0.99 | 0.99 | 1478 |
| 13 | 0.96 | 1.00 | 0.98 | 1454 |
| 14 | 0.99 | 0.99 | 0.99 | 4383 |
| 15 | 0.99 | 0.99 | 0.99 | 1513 |
| 16 | 0.97 | 0.98 | 0.97 | 512 |
| 17 | 0.99 | 0.97 | 0.98 | 977 |
| 18 | 0.99 | 1.00 | 0.99 | 3693 |
| 19 | 0.99 | 0.99 | 0.99 | 1740 |
| 20 | 0.99 | 0.99 | 0.99 | 2256 |
| 21 | 0.98 | 0.98 | 0.98 | 369 |
| 22 | 0.99 | 0.97 | 0.98 | 907 |
| 23 | 0.97 | 0.99 | 0.98 | 500 |
| 24 | 0.99 | 0.98 | 0.98 | 875 |
| 25 | 0.99 | 0.99 | 0.99 | 539 |
| accuracy | | | 0.99 | 30165 |
| macro avg | 0.98 | 0.98 | 0.98 | 30165 |
| weighted avg | 0.99 | 0.99 | 0.99 | 30165 |

# Conclusion

- Convolutional neural networks are multi-layer neural networks that are really good at getting the features out of data. They work well with images and they don't need a lot of pre-processing.
- Using convolutions and pooling to reduce an image to its basic features, you can identify images correctly.
- It's easier to train CNN models with fewer initial parameters than with other kinds of neural networks. You won't need a huge number of hidden layers because the convolutions will be able to handle a lot of the hidden layer discovery for you.
- One of the cool things about CNNs is the number of complex problems they can be applied to. From self-driving cars to detecting diabetes, CNNs can process this kind of data and provide accurate predictions.

# References


- https://www.freecodecamp.org
- https://machinelearningmastery.com
- https://machinelearningknowledge.ai
- https://en.wikipedia.org/
- https://careerfoundry.com
- https://www.sciencedirect.com
- https://developers.google.com/
- https://computersciencewiki.org
- https://www.tensorflow.org/
- https://towardsdatascience.com
- Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition Dominik Scherer, Andreas Müller and Sven Behnke
- https://www.kaggle.com
- https://keras.io
- https://stackoverflow.com/

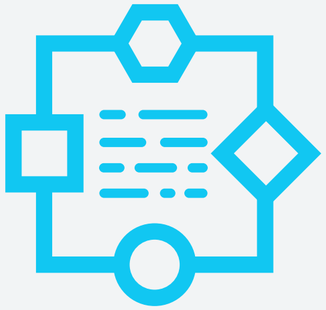
TECH ACADEMY

# Our Team

Yasin
Vəliyev

Vüsal
Həsənov

Azər
Allahverdiyev



TECH
ACADEMY

# Thank's For Watching