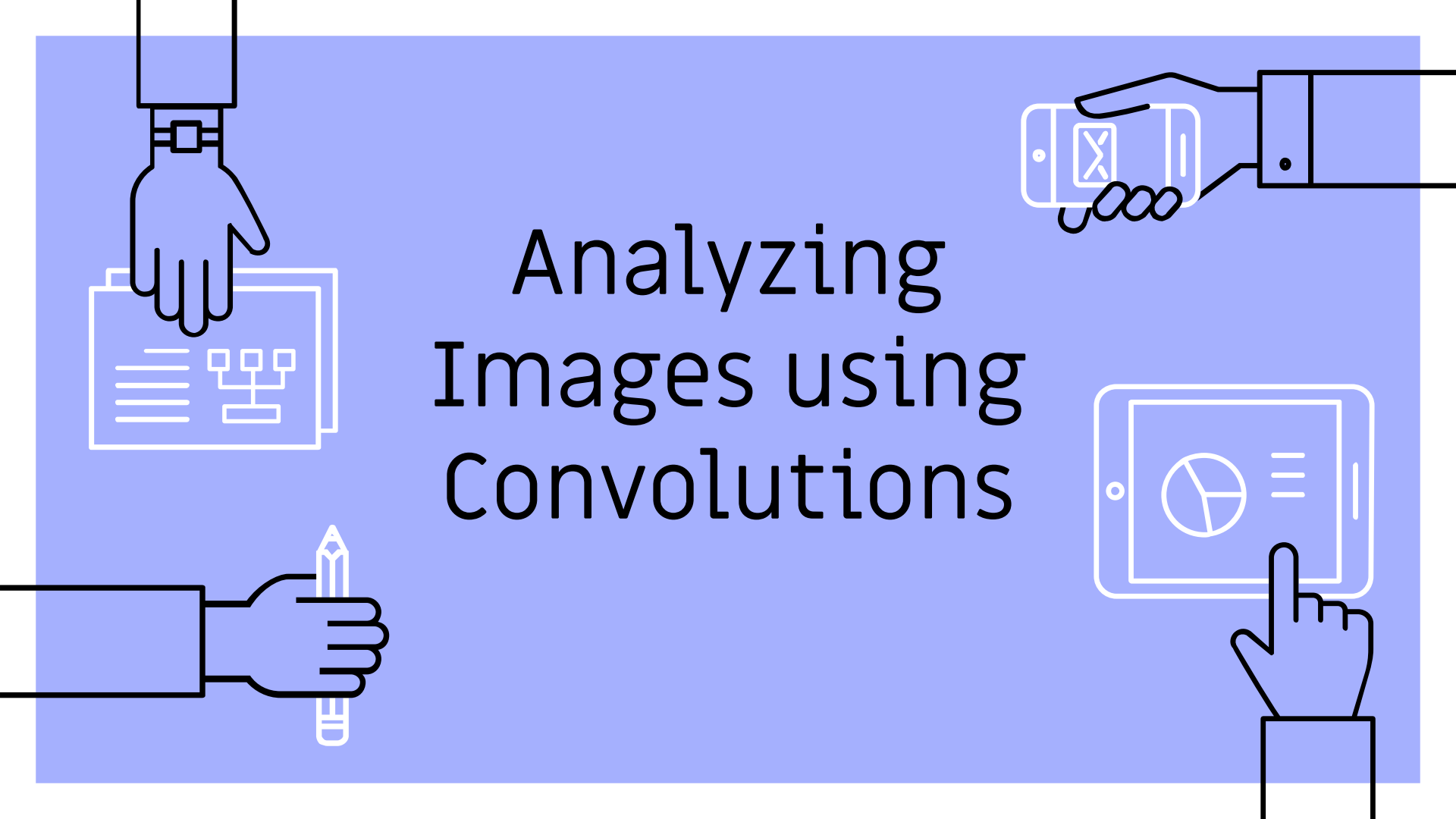
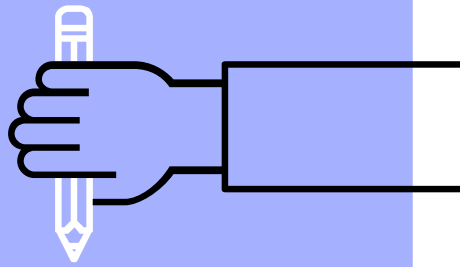
Analyzing Images using Convolutions

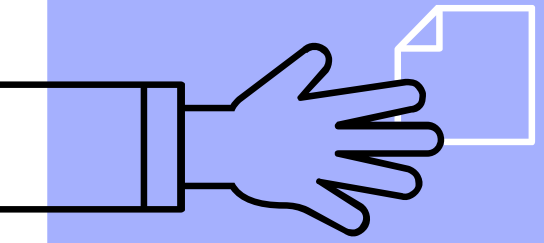


AGENDA

1. BACKGROUND
2. BUILD MODEL
3. OPTIMIZATION
4. SUMMARY

1.
BACKGROUND





BACKGROUND

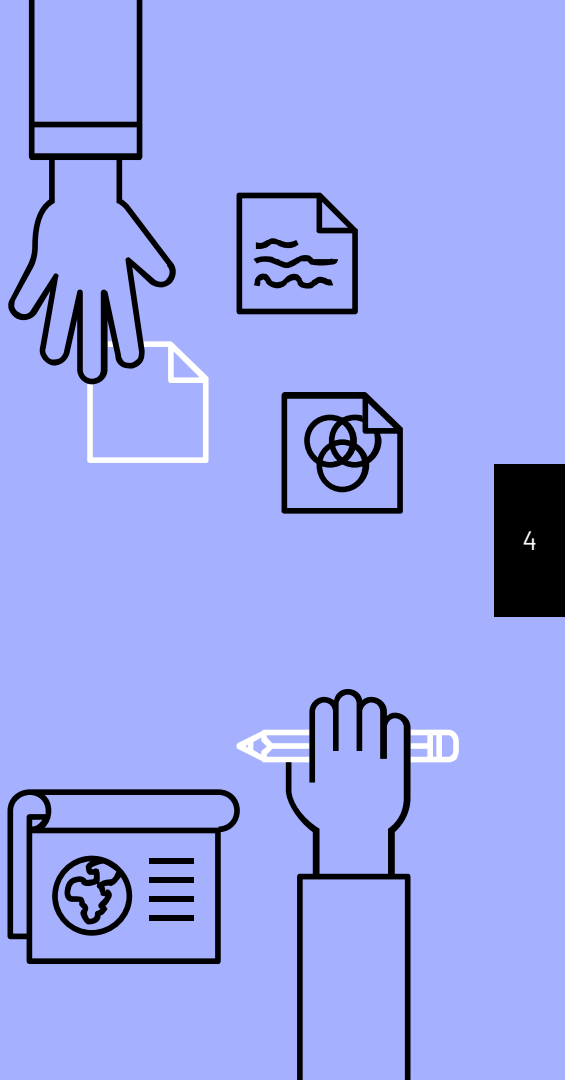
- DATA:

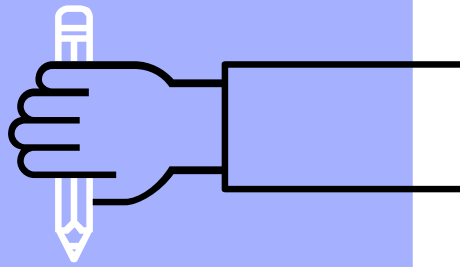
Fashion-MNIST - a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples

- GOAL:

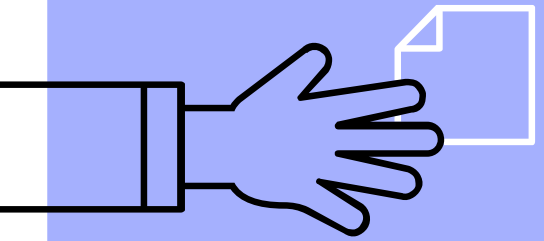
Build the best possible model to predict the class of the image as measured on the test set

- TEAMWORK:

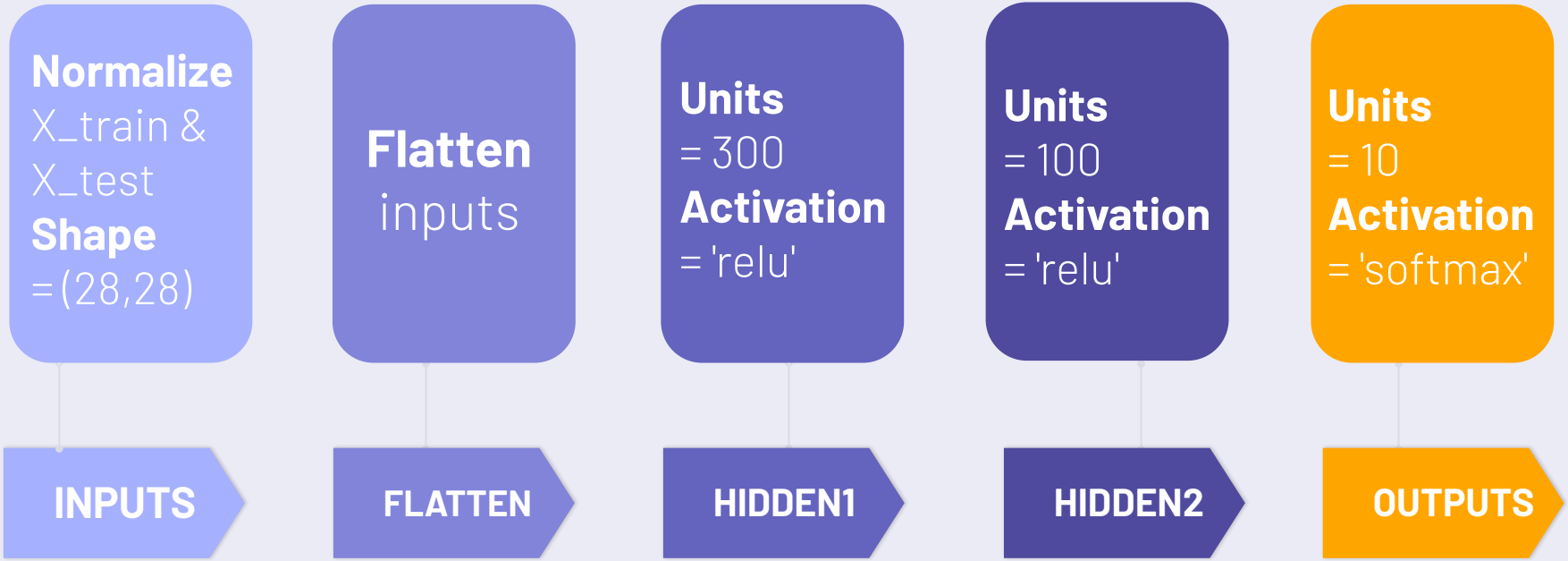




2.
BUILD MODEL

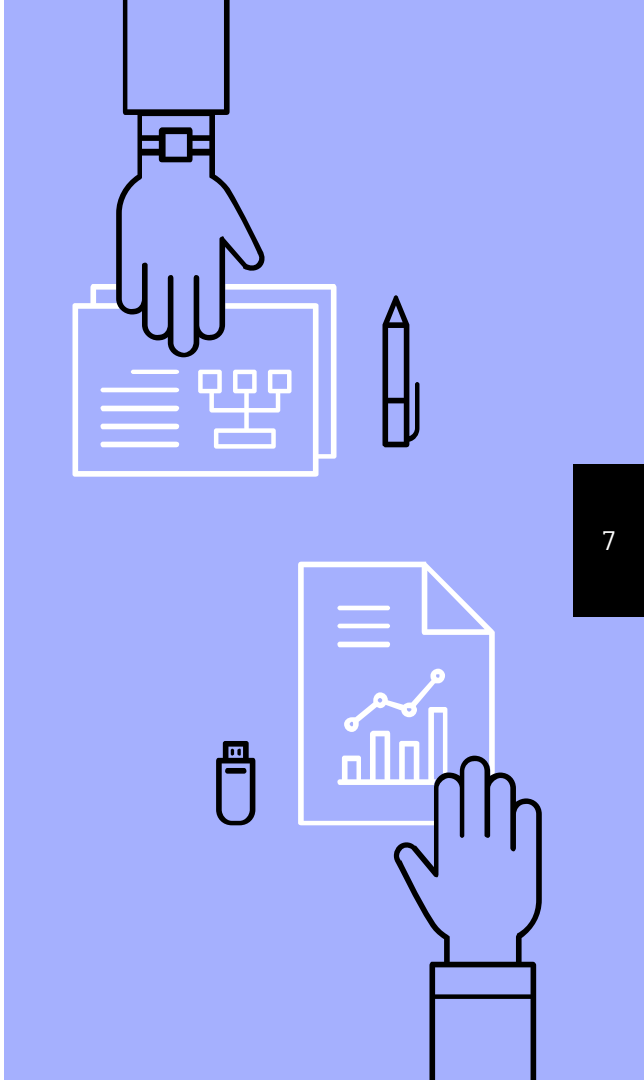


Baseline Model - Architecture



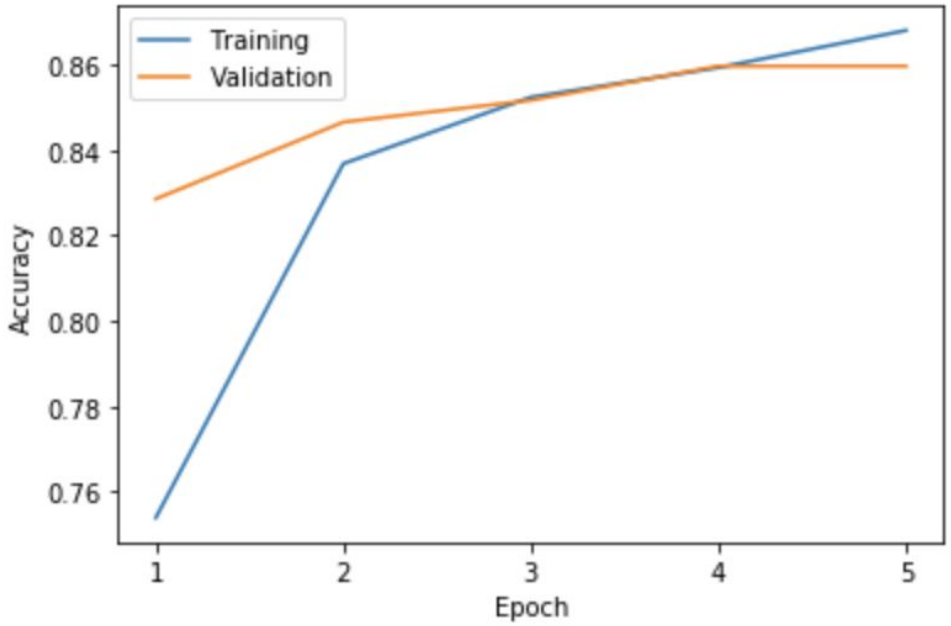
Baseline Model - Compile and fit model

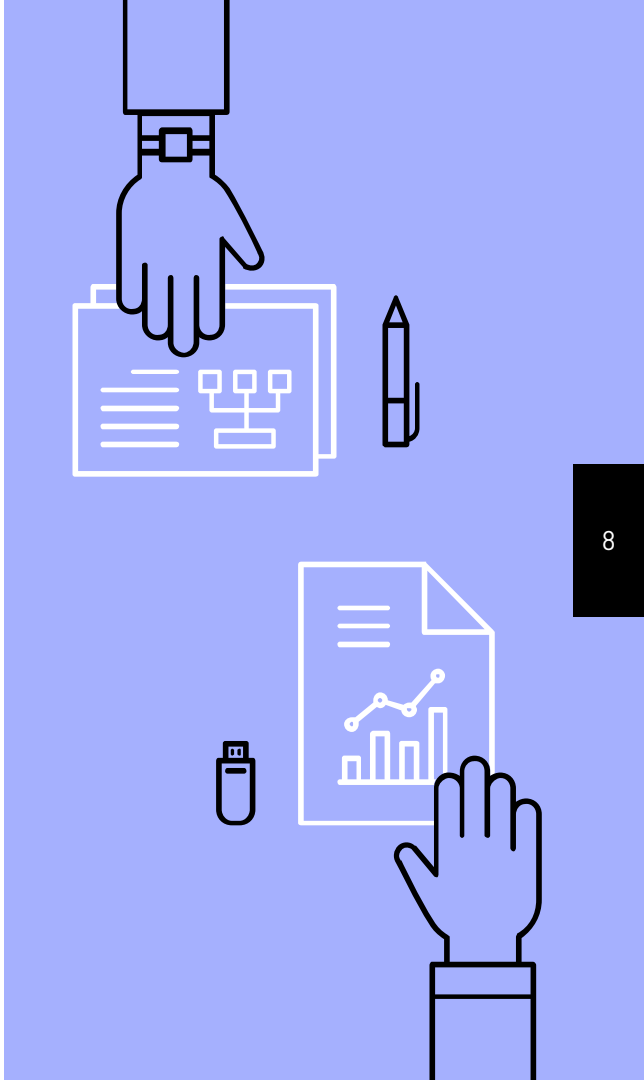
- **Loss** = 'sparse_categorical_crossentropy'
- **Optimizer**=SGD
  Learning rate =0.01
  Momentum=0.9
- **Training dataset**: X_train[:50000]
- **Validation dataset**: X_train[50000:]
- **Batch_size** = 200
- **Epochs** = 5



Baseline Model - Result

Accuray for test set = 0.8512





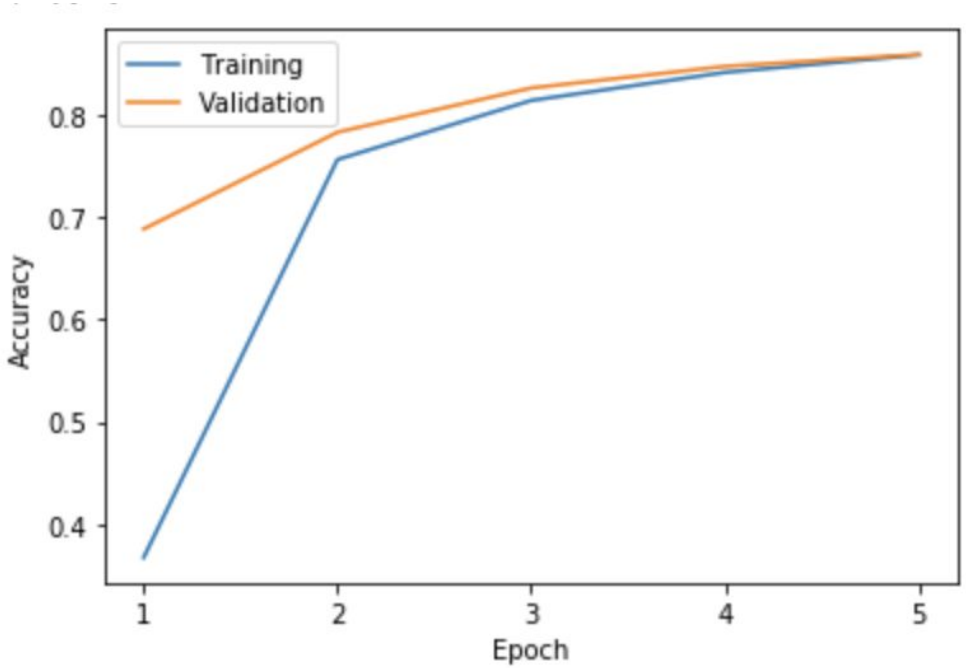
3.
OPTIMIZATION

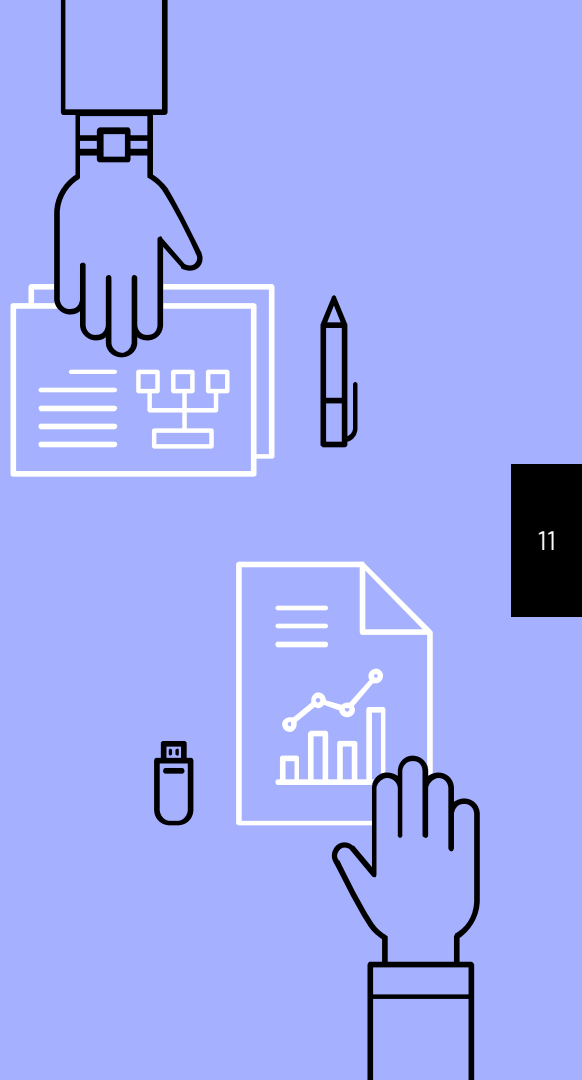
OPTIMIZATION - Add autoencoder

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 28, 28)]          0
_________________________________________________________________
encoder (Model)              (None, 3, 3, 64)          23296
_________________________________________________________________
decoder (Model)              (None, 28, 28)            23233
_________________________________________________________________
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 300)               235500
_________________________________________________________________
dense_1 (Dense)              (None, 100)               30100
_________________________________________________________________
out (Dense)                  (None, 10)                1010
=================================================================
Total params: 313,139
Trainable params: 313,139
Non-trainable params: 0
```

Add autoencoder - Result

Accuray for test set = 0.8575





OPTIMIZATION
- Increase epoch and add early stop

➔ Based on the autoencoder architecture

➔ Increase epoch:

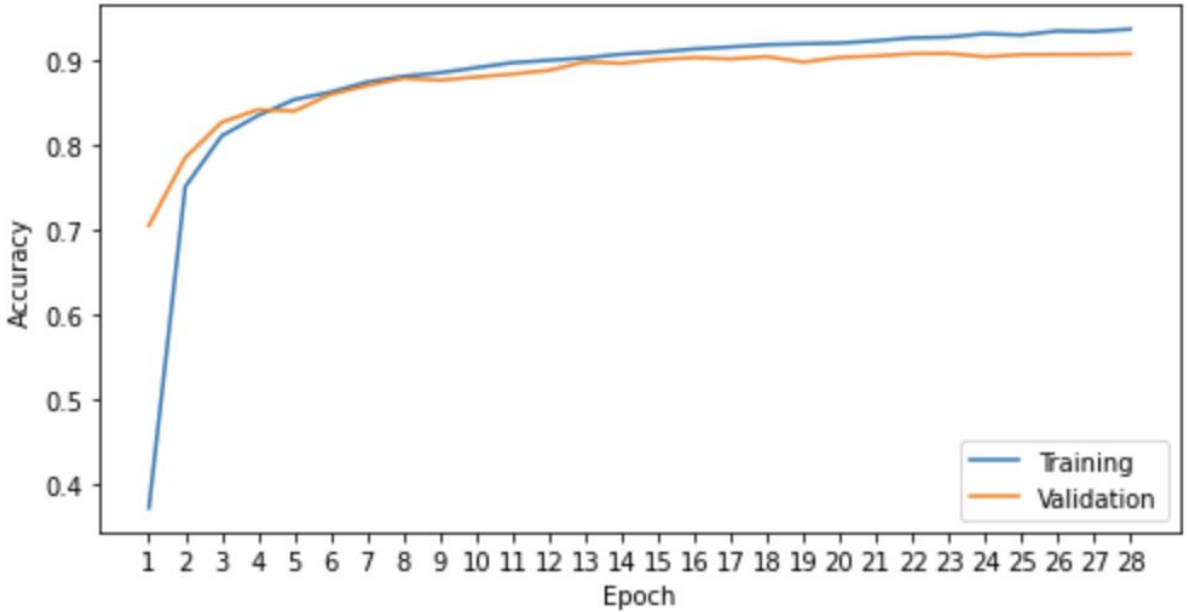
Epochs = 50

➔ Add early stopping:

Patience = 5

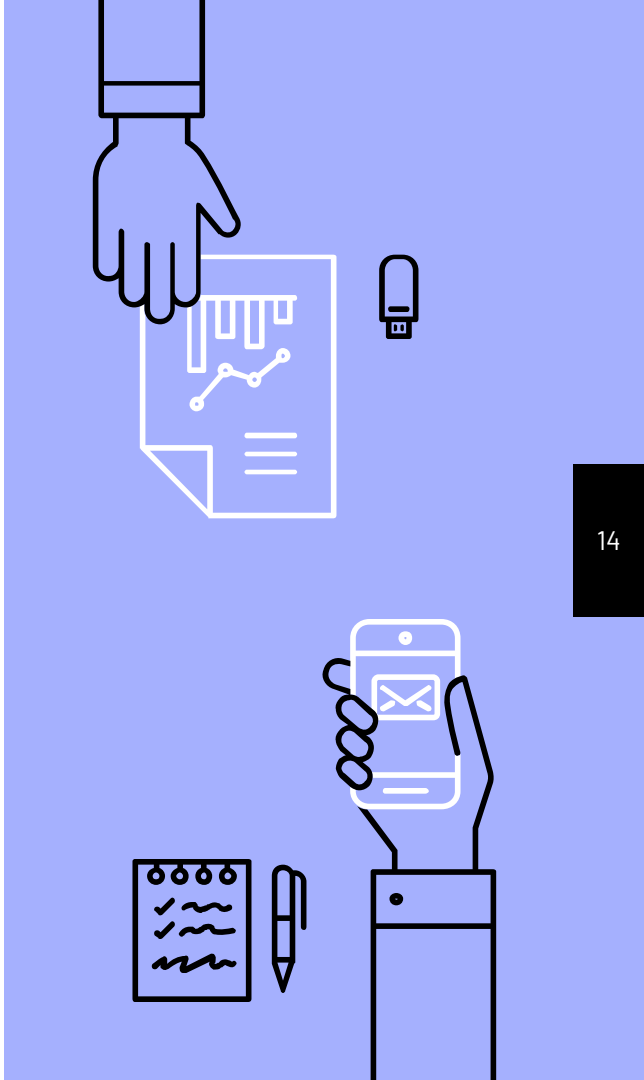
Increase epoch and add early stop - Result

Accuray for test set = 90.33%

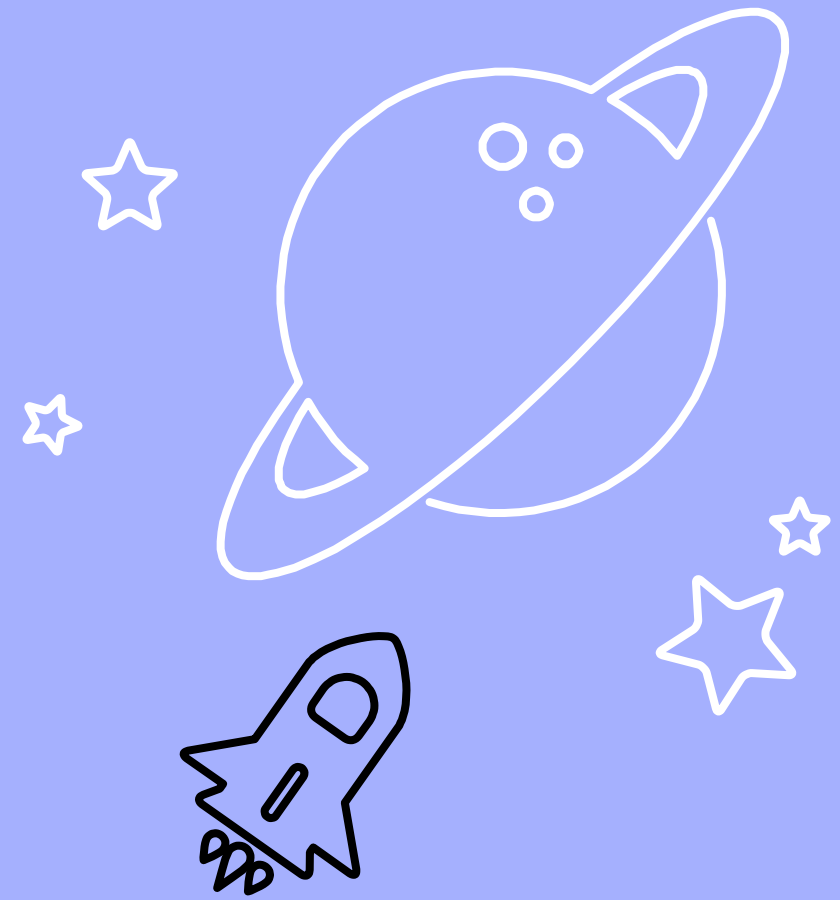


Accuracy Comparison

Models	Accuracy on test set
Baseline Model	0.8512
Add Conv2D layers	too slow to run
Add autoencoder	0.8575
Increase epoch and add early stop	0.9033



4.
SUMMARY



Summary

Conv2D layers

Adding Conv2D layers and increasing filters may increase the prediction accuracy, but it took too long to run

Autoencoder

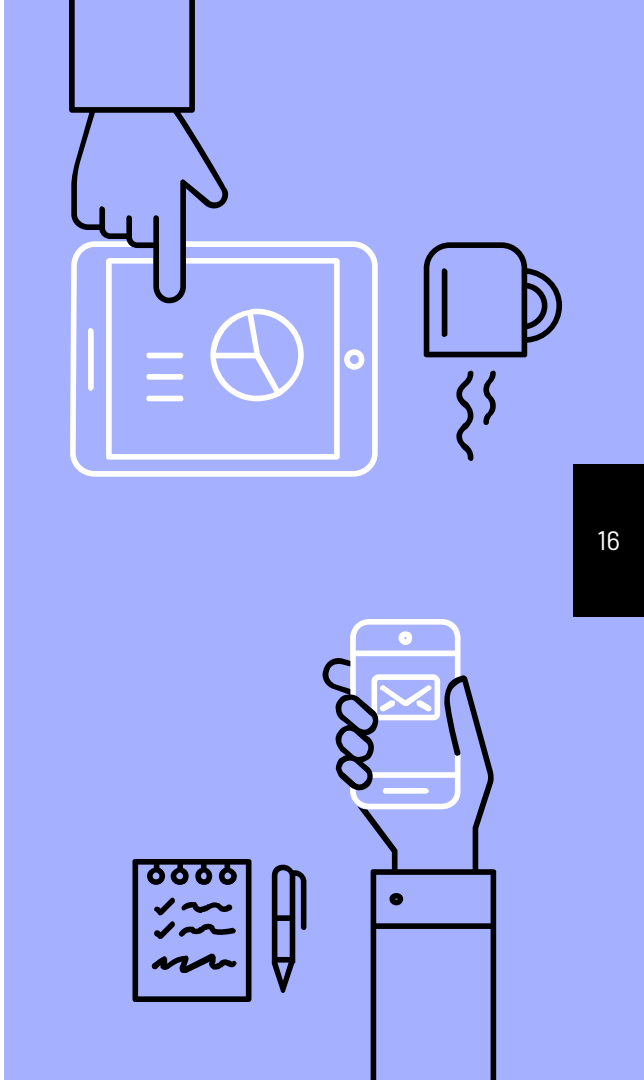
Adding autoencoder with Conv2D **slightly increased** the prediction accuracy.

Increase Epoch with Early Stopping

Increasing epochs with early stopping to the previous model **increased** the prediction accuracy **significantly** from 0.8575 to 0.9033

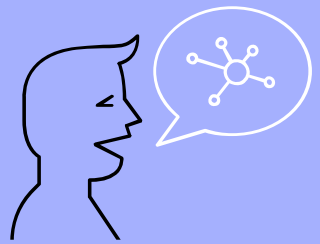
Optimal Model

We got the optimal model **with autoencoder and Conv2D and early stopping (at epoch 23)**, achieving **90%** accuracy on test set.



THANKS!

Any questions?



Credit : Presentation template by SlidesCarnival Photographs by Unsplash

```
####### 2) Add Conv2D layers #######
# Create architecture
inputs = tf.keras.layers.Input(shape=(28,28,1), name='input')
# Conv2D layer
x = tf.keras.layers.Conv2D(filters=64, kernel_size=3, strides=1, padding="same",
activation="relu")(inputs)
x = tf.keras.layers.Conv2D(filters=64, kernel_size=3, strides=1, padding="same",
activation="relu")(inputs)
x = tf.keras.layers.MaxPooling2D(pool_size=2, strides=2, padding="valid")(x)
x = tf.keras.layers.Conv2D(filters=128, kernel_size=3, strides=1, padding="same",
activation="relu")(x)
x = tf.keras.layers.Conv2D(filters=128, kernel_size=3, strides=1, padding="same",
activation="relu")(x)
x = tf.keras.layers.MaxPooling2D(pool_size=2, strides=2, padding="valid")(x)
x = tf.keras.layers.Conv2D(filters=256, kernel_size=3, strides=1, padding="same",
activation="relu")(x)
x = tf.keras.layers.Conv2D(filters=256, kernel_size=3, strides=1, padding="same",
activation="relu")(x)
x = tf.keras.layers.MaxPooling2D(pool_size = 2, strides = 2, padding = "valid")(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(500, activation = 'relu')(x)
x = tf.keras.layers.Dense(250, activation = 'relu')(x)
outputs = tf.keras.layers.Dense(10, activation='softmax', name='out')(x)
model = tf.keras.Model(inputs=inputs, outputs=outputs)
print('Add Conv2D layers')
model.summary()
```