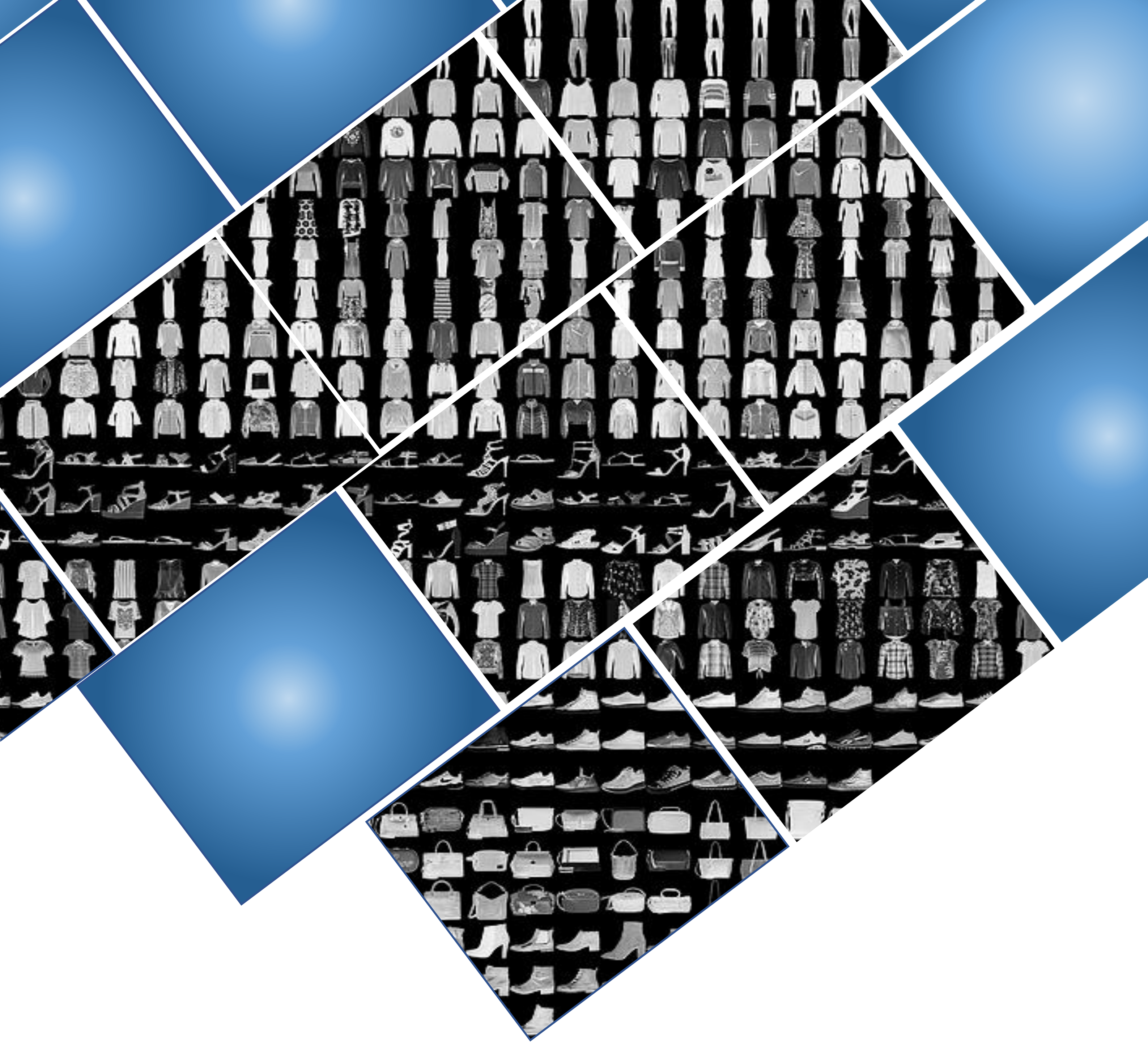


Fashion Items Image Search

Visual search service with Neural Network

Juan Santa Teresa Gómez | KSCHOOL 23rd Master in Data Science | May 2021

Master's Final Project

TABLE OF CONTENTS

1. ABSTRACT

This Final Project is an exercise of Image Classification and Image Search relied of fashion items images, which have been used for training and testing Deep learning.

Data are obtained from Zalando's Research MNIST dataset consisting in 60k labelled examples of 28x28 px and 10 categories.

Different Deep Learning Classification Models has been created from scratch using Keras API and tested their performance iterating in the process of improving accuracy, reaching a value of around 90%.

Production model, including its frontend has been developed using Streamlit API, which allows user to quickly use the search tool.

This document extend the information of the development and for further explanatios of the code, please, see the development notebooks in github repo:

https://github.com/juansantateresa/Fashionitems_imagesearch

2. INTRODUCTION

Deep learning is one of the fields with more development and possibilities in Data Science field. As a field of Machine Learning discipline, Deep Learning covers operations using Artificial Neural Networks, which are inspired by the way brain function develops itself.

A. IMAGE CLASSIFICATION

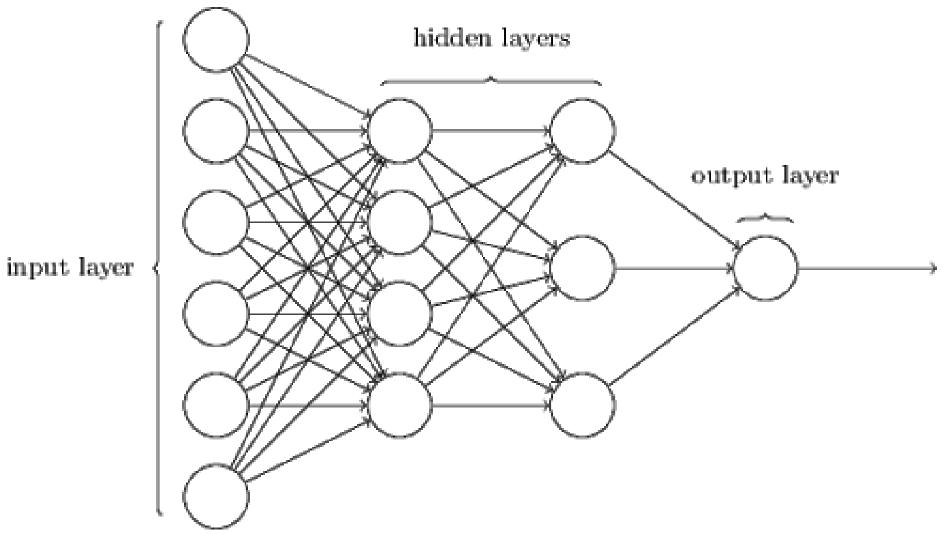
There is a class of Deep Learning Neural Networks commonly used in Image recognition making possible classification and object identification. This Neural Networks are Convolutional Neural Networks.

B. HOW CNN WORKS?

Briefly, CNN works in Image Classification adding weights to certain objects within the image, in that way, once the CNN receives an input image, it returns a convolved feature which is function of the proprieties of it, mostly of the dimension of the kernel.

Repeating this process along the algorithm CNN can extract the high-level features of the image, as edges, background, color or orientation.

CNN Model is the addition of several layers connected making a process throughout which we can, in this use case, extract the features of the imageset and apply them for the purpose.



3. PROJECT BACKGROUND

A. PROJECT AIM

The aim of current project is to develop a Deep Learning model from scratch within a frontend app which allow user to search similar images of fashion items.

To achieve that, current project will cover the successive activities for develop appropriate algorithm to put into production a visual search service, including data preprocessing, performance evaluation and app deploy.

B. PROBLEM SOLUTION

Due to the evolution of Deep Learning libraries, mostly along past decade, several applications became easier to put in practice. One of them is *image search feature* mostly applied to ecommerce. Big retailers such as Amazon, Nike or Zalando recently implemented visual search which allows user to upload a photo or screenshot and search identical or similar product.

The biggest advantage of using visual search from the user perspective is that we process 60.000 times faster visual information than text information, so it makes sense to 'communicate' between data using image than text.

On the other hand, visual search allows shop outfits and looks, instead of certain piece of apparel, which is a disruption in ecommerce growth features.

C. DATA

https://github.com/zalandoresearch/fashion-mnist

Imageset is obtained from Zalando Fashion MNIST which is a dataset provided directly from Zalando for benchmarking machine learning algorithms purposes and, mainly, to supply an alternative to Numbers MNIST.

Imageset contains 60.000 pictures of 28x28 px, distributed in 10 categories. Zalando Research collected performance of several DL classifiers obtaining accuracy between 0.876 and 0.967 which is a good reference in to evaluate performance of current project accuracy.

D. TECHNOLOGY

Obviously, Python Ecosystem, that is to say, Python kernel, libraries as Pandas for managing dataframes and mainly numpy due to the array traduction of images, is used in current project but with some particularities for this use case:

For Deep Learning algorithm this project uses Keras python API which is reasonably usable and provides a fast-paced environment to develop Deep Learning models in a simplier way.

Keras could execute over different libraries, but Google TensorFlow inherit its technical support and currently is it official high level API for Modelling Neural Networks.

On the other hand, the choosen development envirorment is Google Colab, mainly because of taking advantage of features as:

- Virtual environment which offers the chance to work in projects whatever workstation you are using.
- Package installation on the go.
- Easy connection with google drive.
- Possibility to change envirorment to TPU or GPU.

To managing control version and cloud storage of present project github repository will be used for that:

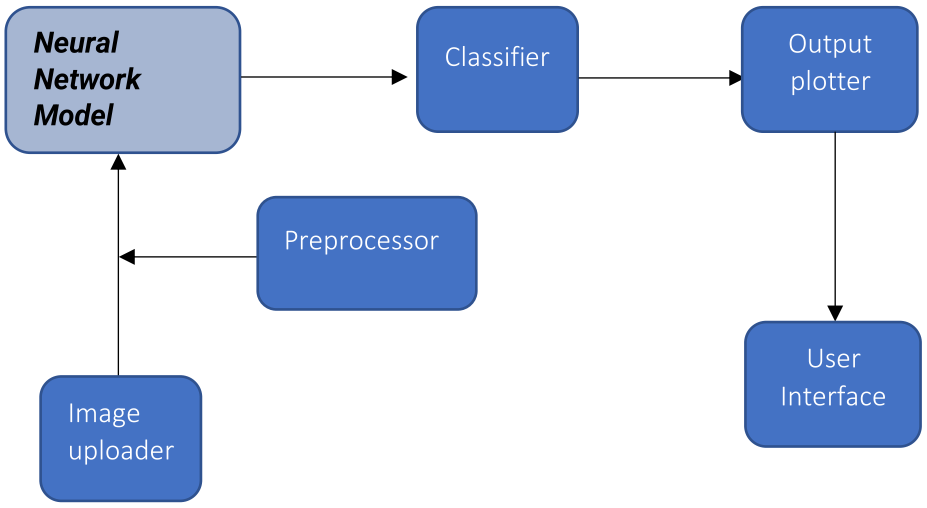
https://github.com/juansantateresa/Fashionitems_imagesearch

In it, the reader can find a readme explaining of development and deploy operations to run the app, but this be explained on detail within following chapters of this project memo.

4. SPECIFICATION AND DESIGN

Going deeper into detail of Project specs and design, firstly, it is appropriate to differentiate main elements, function or scripts which generates the desirable output on the app.

The essential structure for software to provides visual search feature could be the following:



Likewise, Neural Network Model has its own structure, being engaged to a process of design, improvement and input preprocessing which is evaluate under the key metric monitoring.

A. NEURAL NETWORK MODEL

One of the main targets of this project is to explore the neural network creation from scratch though there are some models available already pretrained whereby deploy a model is faster and with guaranteed good performance in image classification.

This condition is chosen in present project is because of balancing the chance of going deeper into Sequential models and testing Keras API being detrimental in the possibility of deploying models faster.

However, pretrained models that could be used in this project are, for instance, the following:

- VGG16
- Resnet50

This project used VGG16 structure as example for creating CNN from scratch.

B. IMAGE UPLOADER AND USER INTERFACE

These features will be provided under app development using Streamlit API, which provides methods of uploading files as web service for introducing file to the algorithm later on

C. PREPROCESSOR

Script for image transformation tasks and translate new image according to the specs of classifier, using preprocessing modules as Image, matplotlib and CV2.

D. CLASSIFIER

This Python module is intended for being able to make predictions for new images using the weights of the trained Deep Learning Model.

E. OUTPUT PLOTTER

For plotting similar images, a function who catch predictions and plot them is needed. This function will connect backend, execute the query to the model and plot in frontend Streamlit App.

F. USER INTERFACE

This feature allow user to query searches uploading images and executing search under Streamlit API.

5. IMPLEMENTATION

A. IMAGE CLASSIFIER CONSTRUCTION

Workflow described below is in 02_Basic_CNN_image_classification and 03_CNN_fine_tuning

TECNOLOGY

As mention above, one of main target of present project is build a CNN from scratch, for that purpose the Library chosen is Keras API, and for this case, project is builded using the typical Sequential model workflow:

`Model = Sequential()` -> For initializing Keras Model

`Model.add()` -> For stacking layers

`Model.summary()` -> For printing shapes (output and input) of layers

B. LAYER MODULES

In .add method Keras allow to add layers which are also built in, so you can import those of interest for your construction.

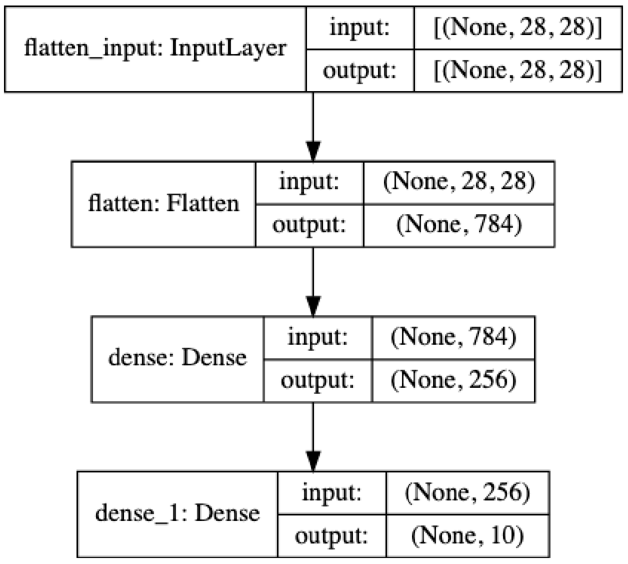
```
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D
```

C. FRAMEWORK

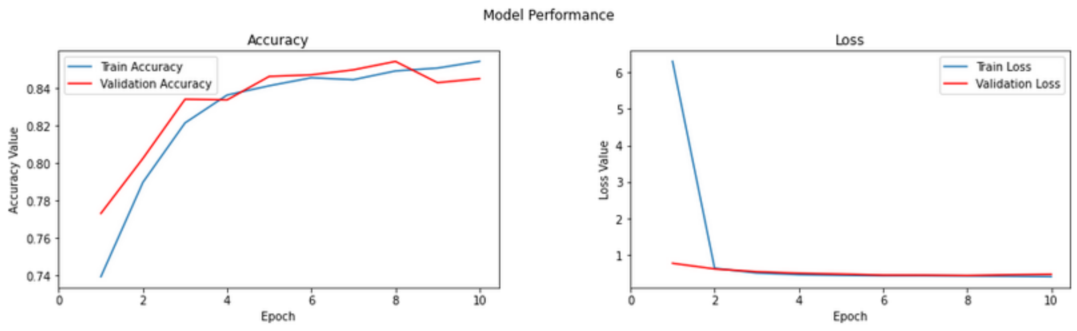
Naïve Model

This project covers the iterative process around of building, testing and measuring performance of the Classifier.

In first instance, a naïve Neural Network (model_1) was built from the ground up only using dense and flatten layers, just for data suitability, as a first approach



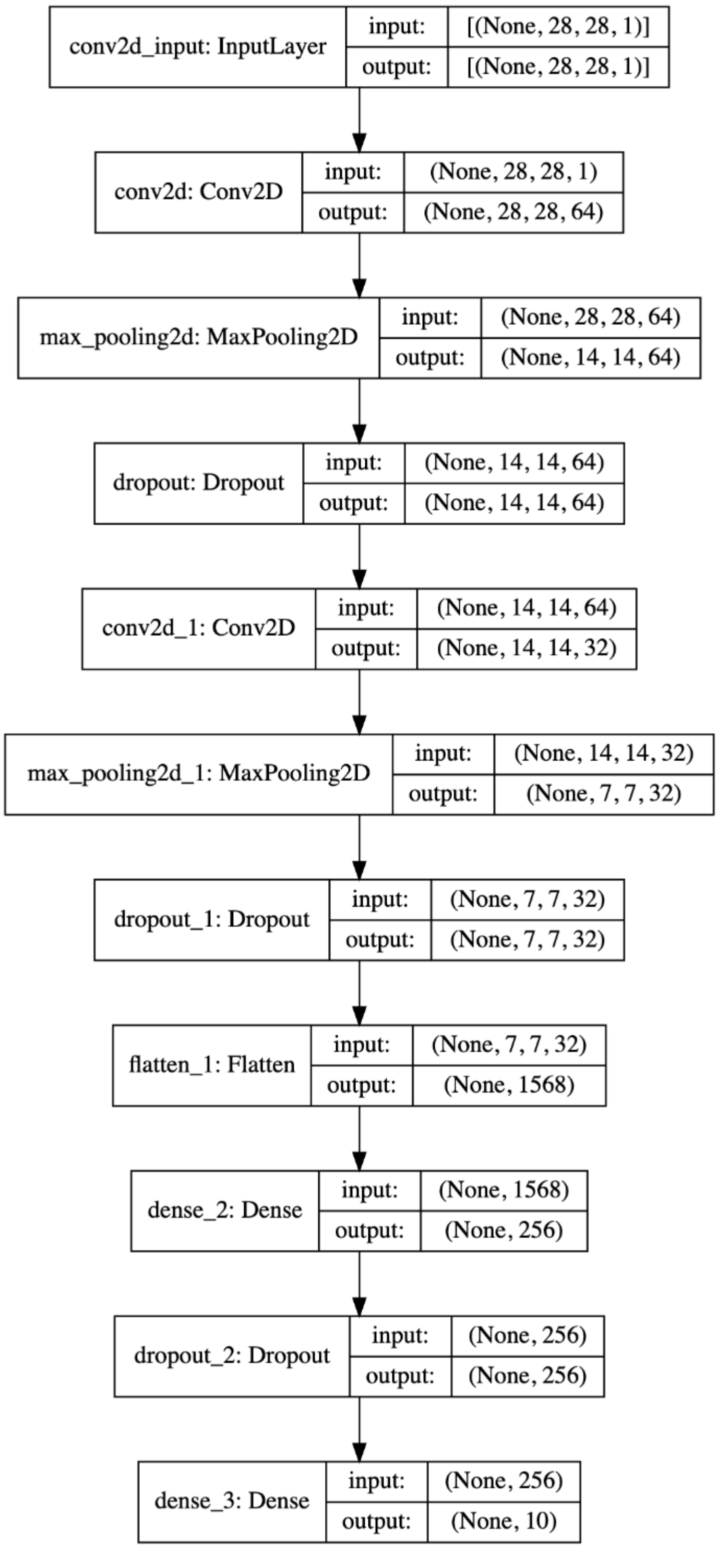
Remarkably, once the model is trained, performance is not entirely bad:



Please note that, there are not even any Conv layer, what is a clear sign of suitability between train and test set and sameness of the images having this first approach performance.

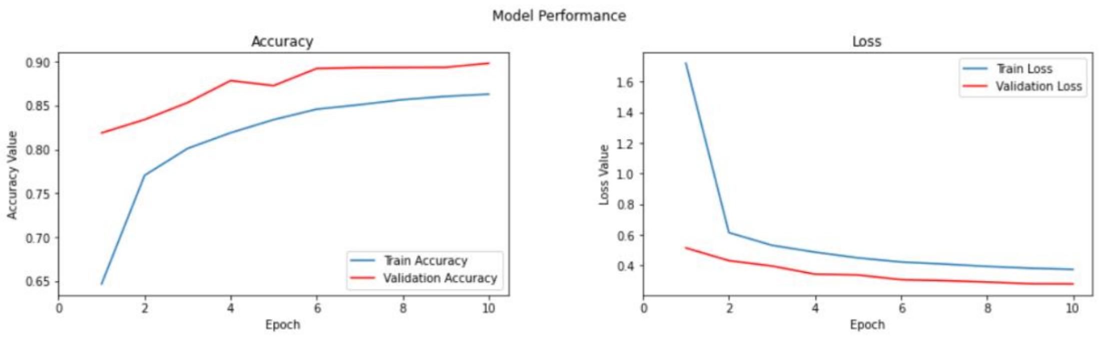
D. FIRST CNN

Further to this first approach, next iteration is to create a CNN (model_") adding Conv2D layers to the model


conv2d_input: InputLayer | input: [(None, 28, 28, 1)] | output: [(None, 28, 28, 1)]
conv2d: Conv2D | input: (None, 28, 28, 1) | output: (None, 28, 28, 64)
max_pooling2d: MaxPooling2D | input: (None, 28, 28, 64) | output: (None, 14, 14, 64)
dropout: Dropout | input: (None, 14, 14, 64) | output: (None, 14, 14, 64)
conv2d_1: Conv2D | input: (None, 14, 14, 64) | output: (None, 14, 14, 32)
max_pooling2d_1: MaxPooling2D | input: (None, 14, 14, 32) | output: (None, 7, 7, 32)
dropout_1: Dropout | input: (None, 7, 7, 32) | output: (None, 7, 7, 32)
flatten_1: Flatten | input: (None, 7, 7, 32) | output: (None, 1568)
dense_2: Dense | input: (None, 1568) | output: (None, 256)
dropout_2: Dropout | input: (None, 256) | output: (None, 256)
dense_3: Dense | input: (None, 256) | output: (None, 10)

The hypothesis underlying this structure is that adding Conv2D will increase performance. On the other hand, just as additional features to convolutional layers, it convenient to add MaxPooling layers for down-sampling the input representation sequentially and Dropout which enhances the generalization over the network, just randomly dropping elements of the input. Also, output keeps as 1D element with Flatten and Dense layers.

Regarding activation, project uses 'relu' in a default way for image classification taking advantage or fast converging and computing efficiency.

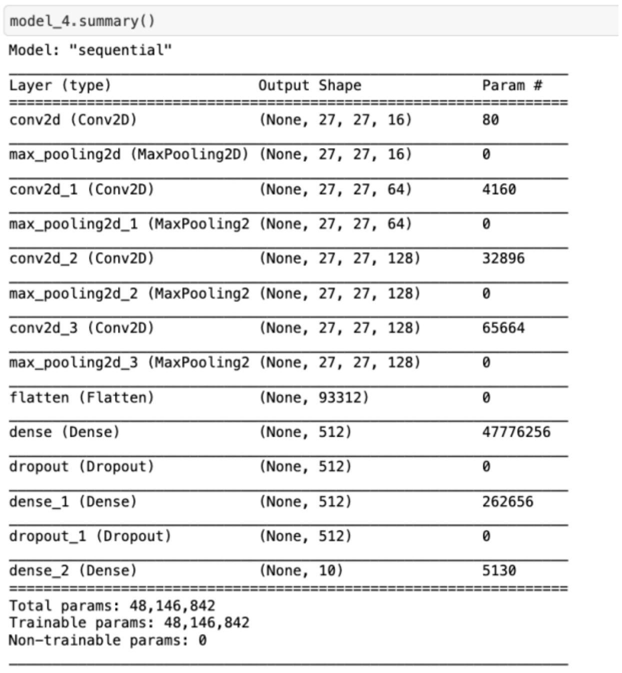


Performance for model_2 is 89.8% for validation accuracy and 27.8% for validation loss.

In this case, concerning model fitting this project uses small quantity of epochs since accuracy seems decent from scratch, taking into account that is mandatory to balance computing resources dedicated for training as we are working on Google Colab.

E. FINE TUNED CNN

Following previous framework, next iteration is improve previous CNN pattern by, basically, adding nodes, more Conv2D layers and increasing epochs for fitting.

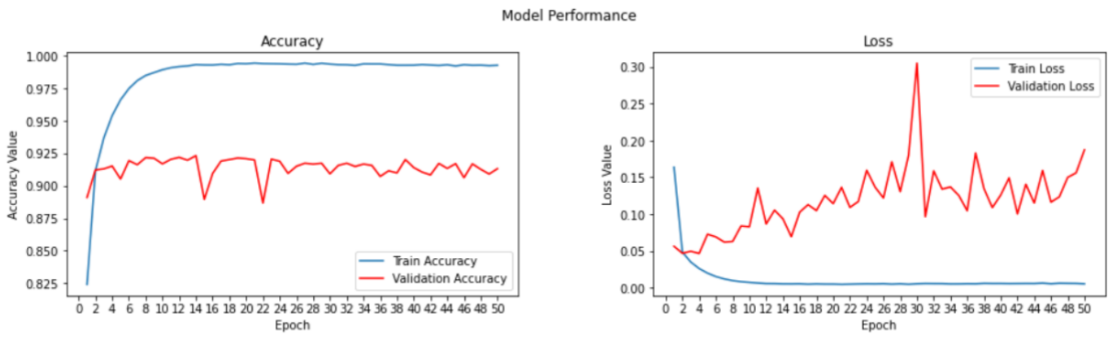


```
model_4.summary()
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 27, 27, 16)        80
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 27, 27, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 27, 27, 64)        4160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 27, 27, 128)       32896
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 27, 27, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 27, 27, 128)       65664
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 27, 27, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 93312)             0
_________________________________________________________________
dense (Dense)                (None, 512)               47776256
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 48,146,842
Trainable params: 48,146,842
Non-trainable params: 0
_________________________________________________________________
```

Accordingly in model fitting computing efficiency is weighed, this way, although epochs have been increased, batch_size is set low.

```
history_4=model_4.fit(Xtrain,ytrain, batch_size=96,
              epochs=50, validation_data=(Xval,yval))
```

Performance of model_4 for validations is 91% and validation loss is 18%, so it will be used in production.



F. DATA ACQUISITION AND PREPROCESSING

All data acquisition and preprocessing workflow is in 01_Data_Adquisition_and_Preprocessing

Fashion MNIST is in 2021 integrated as another MNIST API so can be imported directly. As other of the goals of present project is testing workflow in Google Colab and its connection with google drive or the possibilities to work in virtual environment storage, data is stored in free links of gdrive so it can be imported directly to current Colab session.

G. DATA STRUCTURE

As mentioned above, imageset contains 60.000 pcs and images are 28x28 px and for loading and array traduction along the project several libraries have been used, but mainly matplotlib for showing and PIL Image or Keras preprocessing which is Image package working under this API.

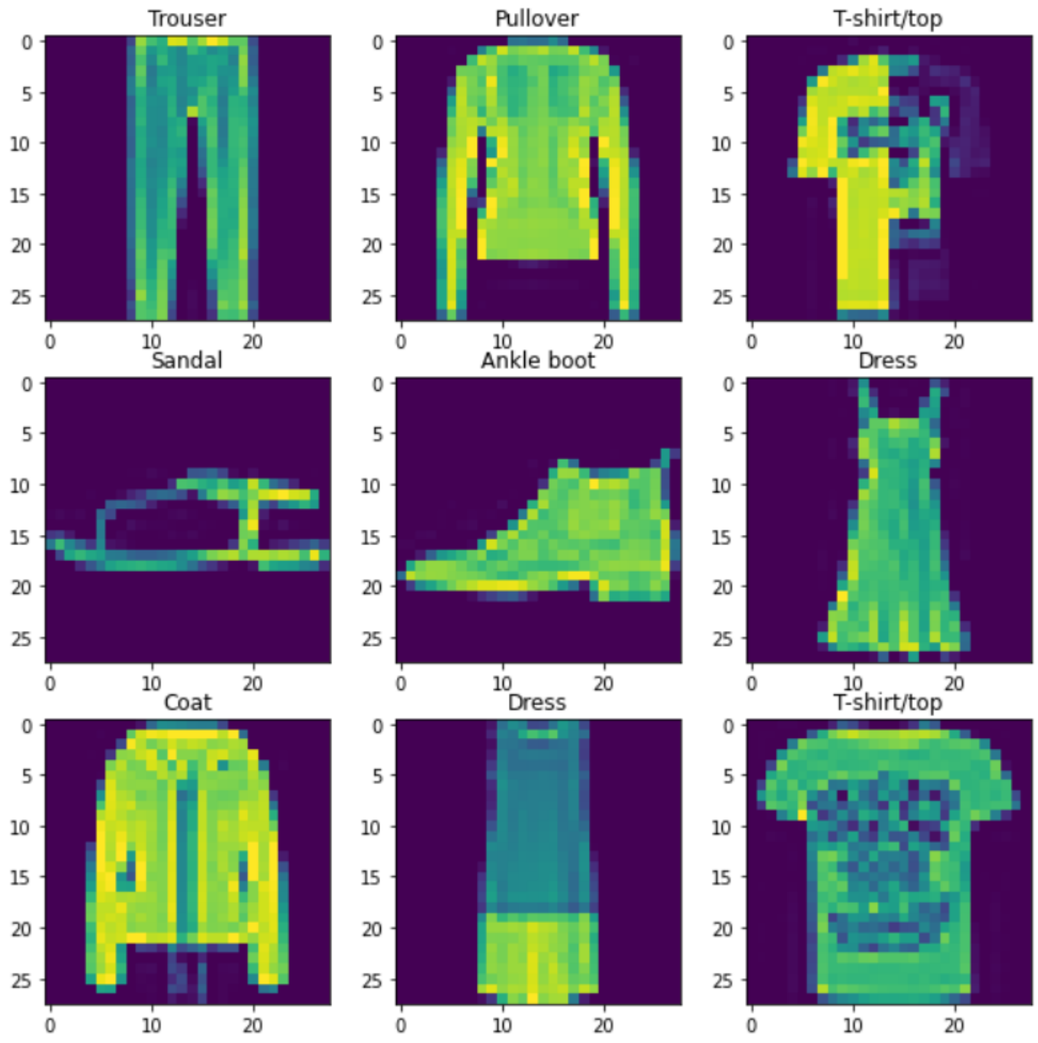
First task in this field is loading and uncompressing raw data and transform it into array, for that purpose function loader (load_imgset) is defined.

Also, a validation dataset is created splitting 6000 pcs of raw dataset.

In order to prepare categories for introducing them into our model is necessary to transform labels into categorical values.

H. DATA EXPLORATION AND PLOTTING

For quickly plotting images and its labels, this project uses plotting function (plot_random_grid) which returns a random 3x3 grid being useful to visual exploration and differentiation of categories.



Once data is ready to be entered in the algorithm, next step is creating the feature extraction process.

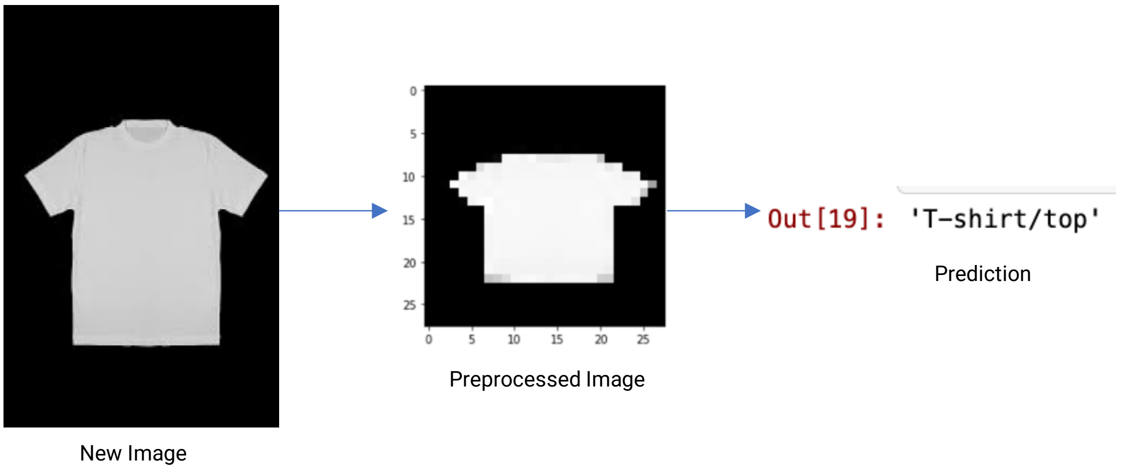
I. FEATURE EXTRACTION/CLASSIFIER

Deploying a visual search service requires a function to extract features of a new input image, for that purpose, once the model_4 is trained, its weights have been stored into .h5 file, saved in gdrive.

That way, once the input image is preprocessed we can cast a prediction using .predict Keras Method. This returns an array of n positions, with n as the number of labels and its probability of being classified as certain label.

```
model_4_bis.predict(sample_img_tensor)
```

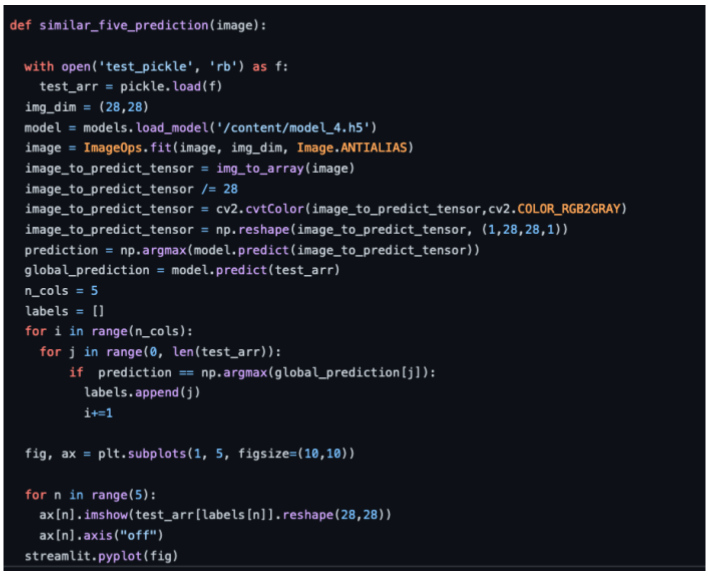
The classifier is applied to new image executing a prediction with success:



```
Out[19]:  'T-shirt/top'
```

J. PREDICTION PLOTTER

Function similar_predict_plot roam all predictions generated within test imageset and catch similar predictions in a certain number than the predicted label of the image.

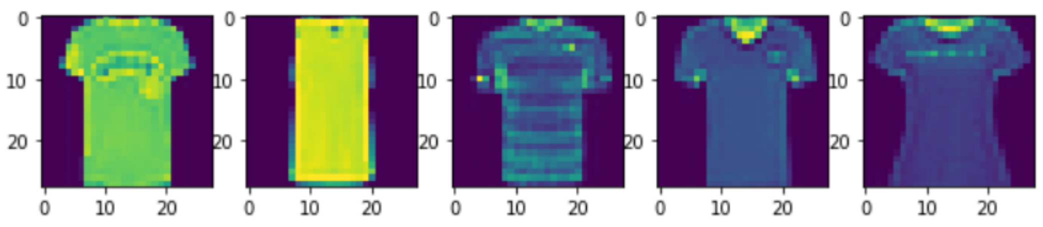


```
def similar_five_prediction(image):

  with open('test_pickle', 'rb') as f:
    test_arr = pickle.load(f)
  img_dim = (28,28)
  model = models.load_model('/content/model_4.h5')
  image = ImageOps.fit(image, img_dim, Image.ANTIALIAS)
  image_to_predict_tensor = img_to_array(image)
  image_to_predict_tensor /= 28
  image_to_predict_tensor = cv2.cvtColor(image_to_predict_tensor,cv2.COLOR_RGB2GRAY)
  image_to_predict_tensor = np.reshape(image_to_predict_tensor, (1,28,28,1))
  prediction = np.argmax(model.predict(image_to_predict_tensor))
  global_prediction = model.predict(test_arr)
  n_cols = 5
  labels = []
  for i in range(n_cols):
    for j in range(0, len(test_arr)):
        if  prediction == np.argmax(global_prediction[j]):
          labels.append(j)
          i+=1

  fig, ax = plt.subplots(1, 5, figsize=(10,10))

  for n in range(5):
    ax[n].imshow(test_arr[labels[n]].reshape(28,28))
    ax[n].axis("off")
  streamlit.pyplot(fig)
```



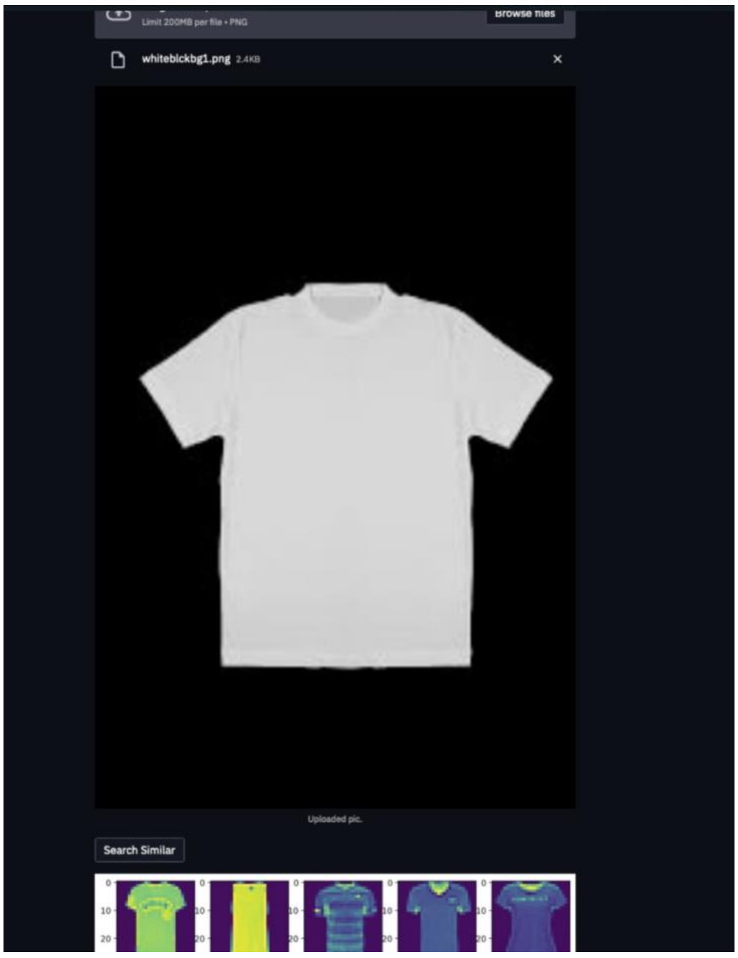
K. FRONTEND

To allow user to query searches, a Streamlit App has been deploy. This module calls the similar predict plot and shows results under Streamlit UI. In this use case functions used are:

Sreamlit.file_uploader

Streamlit.button

Streamlit.pyplot

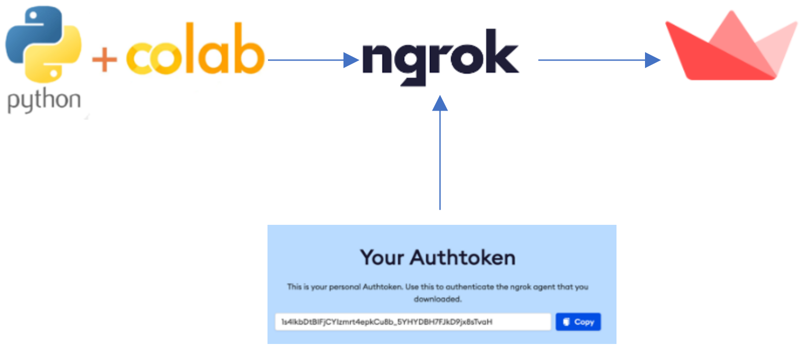


L. APP DEPLOY

As the all-in-one envirorment for this project is Google Colab, to run the App, the runtime envirorment of Google Colab is used for storage and launch the functions.

Streamlit requires a secure tunnel to run an app connected from a localhost to a public url. In this case, nngrok is used for that purpose.

Ngrok requires of a authtoken, under a user account creation, this token allows pyngrok module to be loaded and, afterwards, launch a public url with Streamlit App.



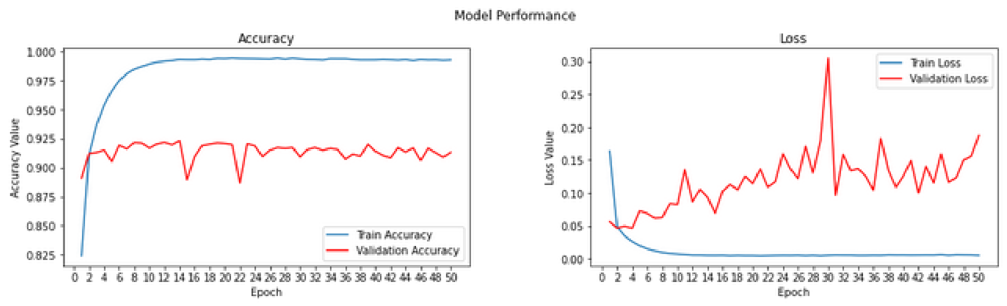
This is your personal Authtoken. Use this to authenticate the ngrok agent that you downloaded.

1s4lkbDtBIFjCYIzmrt4epkCu8b_5YHYDBH7FJkD9jx8sTvaH

6. RESULTS AND EVALUATION

A. VALIDATION ACCURACY AND LOSS

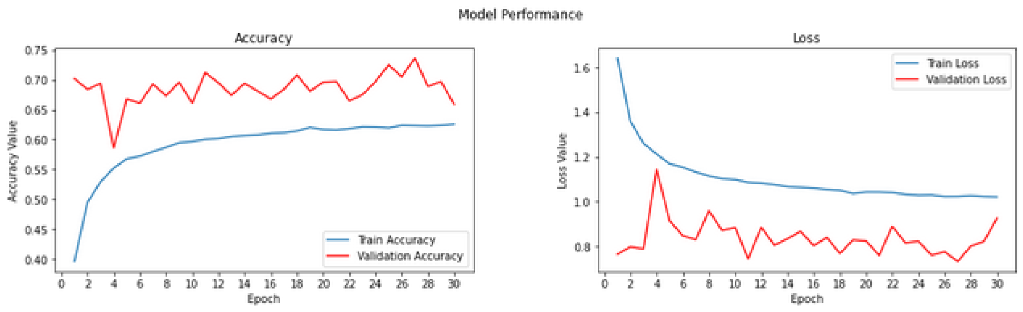
As mentioned above, maximum performance for production model training were the following, 92% for validation accuracy and 18% for validation loss.



B. STRATEGIES TO INCREASE ACCURACY

Image augmentation was tested as a Hypothetical way to increase accuracy due to the sameness of the dataset. For that, current project uses ImageDataGenerator of Keras. Train set generated by ImageDataGenerator is as a result of applying a function which generate variations in a certain batch of the imageset .

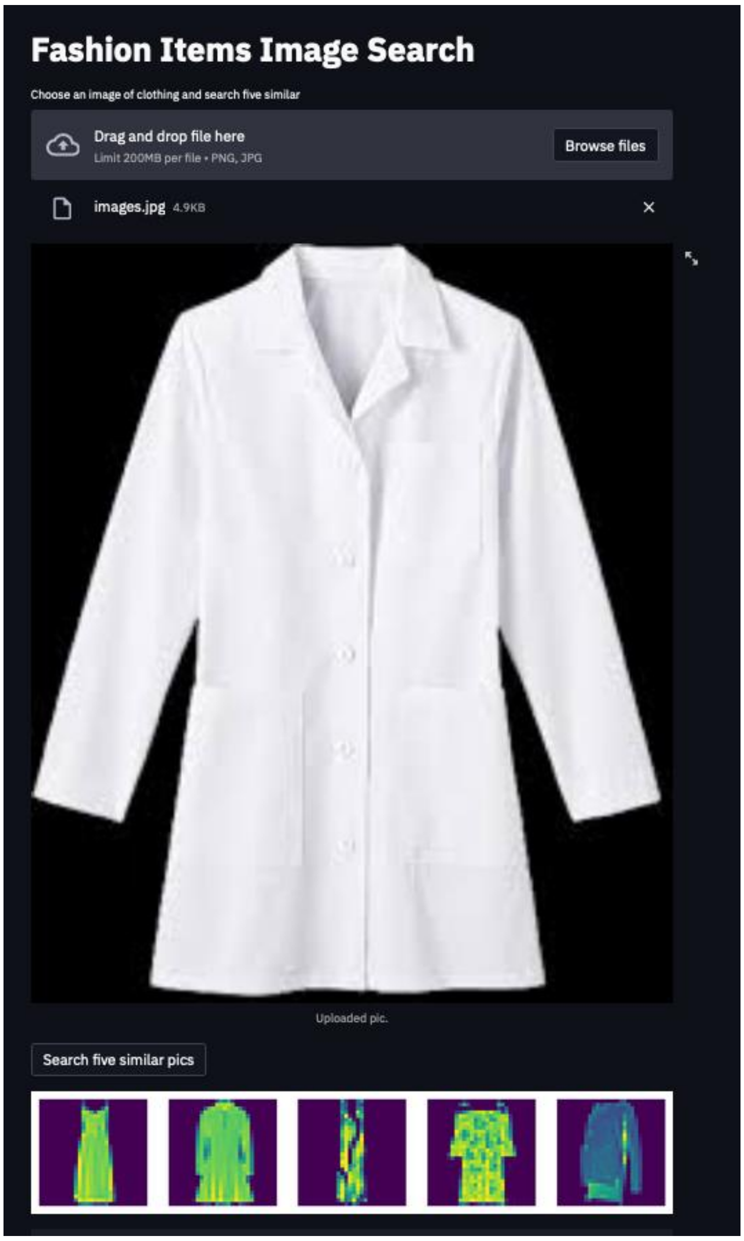
It was introduced in model, but the results shown signs of overfitting:



This way, results of production model (model_4) are within the range of Zalando benchmarks.

App performance

Due to the sameness of the data, similar images are identified and visual search executes with no issue and backend running works within the Colab envirorment.


Fashion Items Image Search
Choose an image of clothing and search five similar
Drag and drop file here
Limit 200MB per file • PNG, JPG
Browse files
images.jpg 4.9KB
Uploaded pic.
Search five similar pics

7. REFERENCES

1. https://arxiv.org/abs/1708.07747 Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Han Xiao, Kashif Rasul, Roland Vollgraf
2. Kschool 23 Master Data Science