Project Report

Gender and Ethnicity Prediction of UTKFaces CV Project

Babiak Chiara (if20b061), if20b061@technikum-wien.at
Führer Angelika (if20b090), if20b090@technikum-wien.at
Rinner Clemens (if20b279), if20b279@technikum-wien.at

Content

Summary

Computer Vision deals with methods for gaining high-level understanding from digital images or videos. Those include for example processing, analyzing and understanding images in order to compute numerical or symbolic information, such as decisions.

We used the principles of CV to create a CNN with the keras sequential model to classify the UTKFaces data according to gender and ethnicity.

The project environment chosen is a jupyter notebook written in the programming language python (ver. 3) and it includes and utilizes the following libraries:

- Pandas
- Numpy
- CV2
- Sklearn
- Tensorflow

Following Links are important for the Project:

- GitHub
- UTKFaces Dataset

Task and Data Description

Task

The main Goal of this project is to predict gender and respectively ethnicity of a given image. Therefore, two models were created and trained so the features predicted are separated from each other. Still, by defining a function that includes both prediction calls, the predicted classes of an image can be retrieved by declaring just its index in the test data while calling this function.

Data

Generally, the Dataset consists of labelled Images, which have the labels of age, gender, ethnicity and date&time of when the Image was collected/added to the Dataset. Images themselves are taken from the internet and are therefore not coming from a private collection or database leading to them being labelled by the DEX algorithm and double-checked by a human annotator.

All in all there are about 20000 images available in the Set which show different poses and facial expressions of various age/gender/ethnicity groupings.

In our case: age, gender and ethnicity are the three most important features for the model and general working conditions of the dataset.

For such a diverse Dataset it is important for it to be balanced enough so that each feature is represented in the same amount according to the size of the Dataset.

Age

Although we did not implement a model for predicting the age, we did investigate the data so that the model could be added easily later.

The age distribution of the data looks as following:

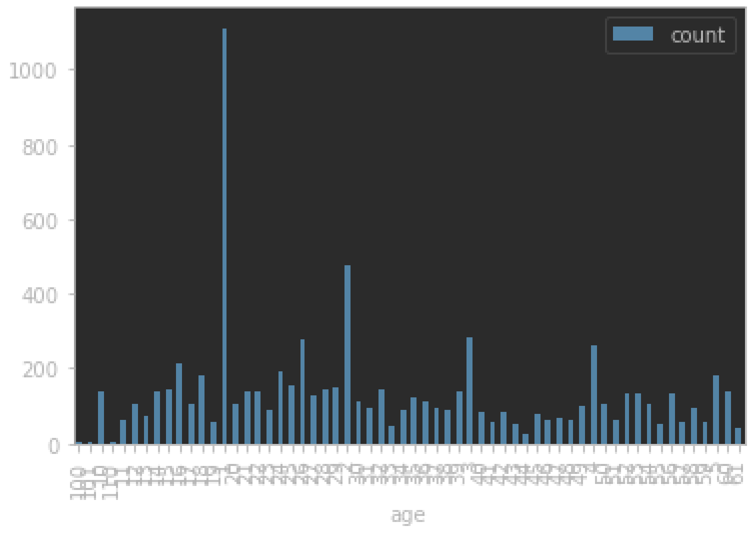


Figure 1: age distribution of the dataset

Since there are too many features in the category age , we have decided to group them together.
The ages range from 0 to 116 years old, to make the grouping easier we oriented them on the stages of life, therefore the following groups have been established:

- Infant = 0-1 year.	0
- Toddler = 2-4 yrs.	1
- Child = 5-12 yrs.	2
- Teen = 13-19 yrs.	3
- Adult = 20-39 yrs.	4
- Middle Age Adult = 40-59 yrs.	5
- Senior Adult = 60+	6

Since we cannot create a pie chart with str and int as values we encoded the age Groups.

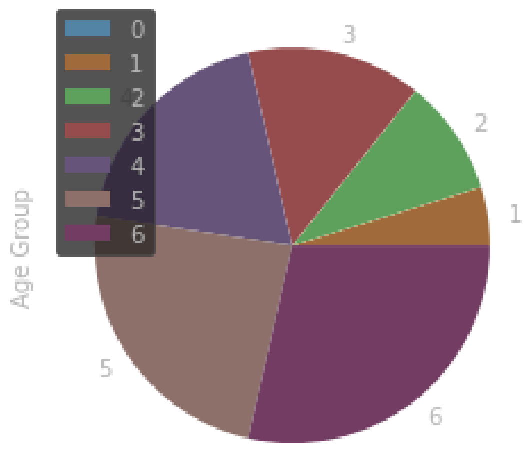


Figure 2: age distribution after grouping

Gender

- We have a slight imbalance between female and male

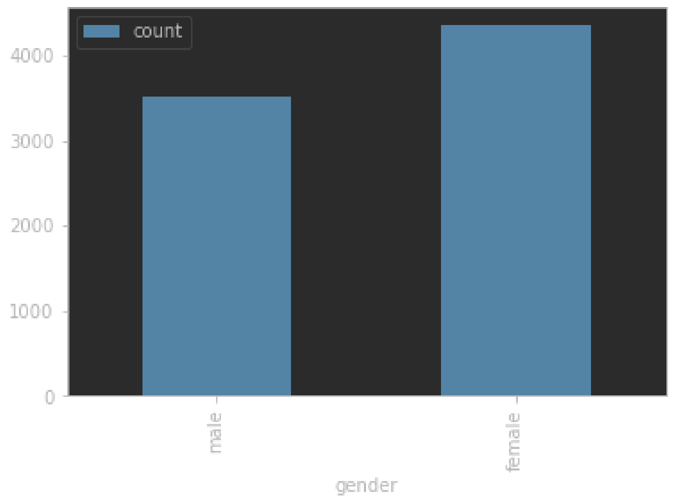


Figure 3: gender distribution of the dataset

Ethnicity

These are the ethnicities we have:

- White
- Asian
- Black
- Indian
- Others

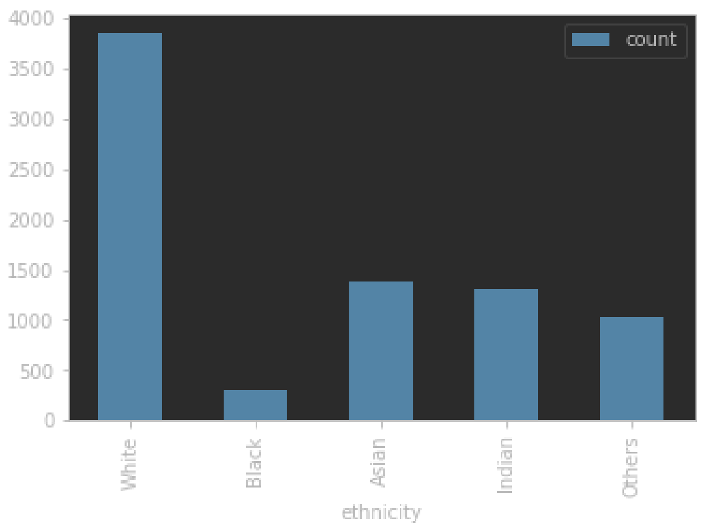


Figure 4: distribution of ethnicity groups

How does the project relate to real applications?

The basis of age, gender and ethnicity detection AI in such applications is face detection technology. After artificial intelligence determines the faces in the image, it analyzes the features of these faces and allows to obtain statistics.

For example, in-store and front-of-shop age detection and gender detection are very effective even in retail businesses. By using these technologies, businesses can identify customer segments, develop marketing strategies according to this segment, and change their product range.

Another possibility is a software which can detect the age range of users passing by a business and suggest advertisements that will affect this age range. Age detection can be used to place ads in the types of media most consumed by your target audience. It is possible to use age detection and gender detection not only for sales purposes but also for security purposes because algorithms can make more accurate age determinations than humans.

Age, gender and ethnicity detection is essential for authentication, behavior analysis, product recommendation based on user preferences, and many other areas. Significant developments have been made in the past few years to meet the need of companies to capture age, gender and ethnicity data. With the introduction of artificial intelligence, the success rate of solving the problem has increased.

Problems encountered and their Solutions

Many of the examples found in the internet which we investigated, used data with a specific folder structure. At the beginning we wanted to use PyTorch but this would have meant much more effort since it is normally used with its integrated DataLoader which needs a dataset split into folders according to the classes. So, we decided to use Tensorflow but still had to split the data by creating lists of each class, defined by the format of the file names.

The labels of each face image are embedded in the file name, formatted like
=> [age][gender][race]_[date&time].jpg

After converting the lists of the data into the right shape, we had no major problems.

Only choosing the right hyperparameters for the Sequential model was a bit tricky because we do not have a lot of previous experience with building a Keras model. However, we know which layers are needed in a neural network for image classification from our course and found several orientation guides.

Description of Code

Task 1: Load Image Dataset

We started by loading the files from the folder "faces/crop_part1/" and split the feature information given by the file name into lists. Later those lists are converted into numpy arrays for further usage.

Task 2: Visualize some of the images

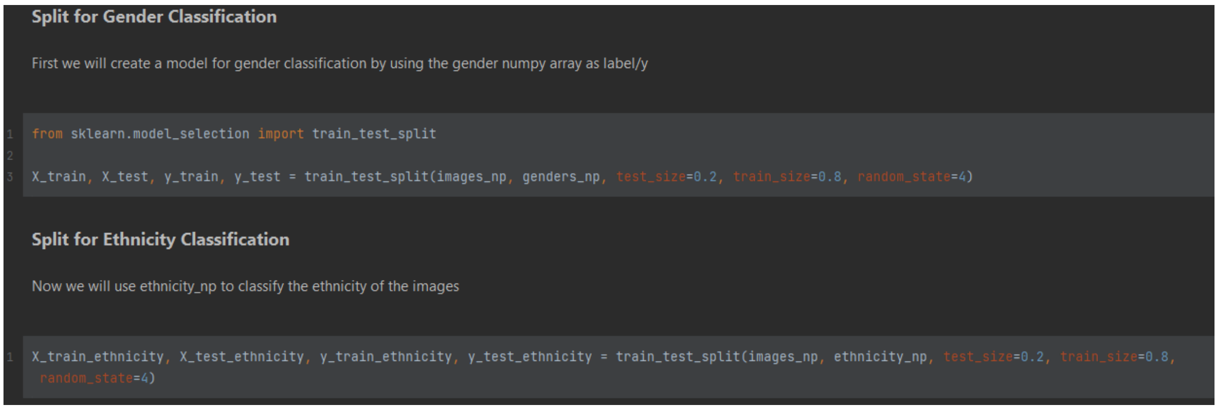
We visualized data from the different features (2-3 pictures) to show what kind of images are representing them as an overview.
The features we have chosen to focus on for the visualization are:

- Age Group
- Ethnicity

Before we did that, we created a dataframe of images for easier handling all the images we have.

Task 3: Train Test Split

Split for each feature by using sklearn.



Split for Gender Classification

First we will create a model for gender classification by using the gender numpy array as label/y

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(images_np, genders_np, test_size=0.2, train_size=0.8, random_state=4)
```

Split for Ethnicity Classification

Now we will use ethnicity_np to classify the ethnicity of the images

```
X_train_ethnicity, X_test_ethnicity, y_train_ethnicity, y_test_ethnicity = train_test_split(images_np, ethnicity_np, test_size=0.2, train_size=0.8,
 random_state=4)
```

Task 4: CNN Model

Gender Classification

The RGB channel values are in the $[0, 255]$ range. This is not ideal for a neural network because the input values should be kept small.

We standardized the values to be in the $[0, 1]$ range by using tf.keras.layers.Rescaling:

```
layers.Rescaling(1./255)
```

For data augmentation we used keras preprocessing layers as well, such as tf.keras.layers.RandomFlip and tf.keras.layers.RandomRotation.

The Sequential model consists of four convolution blocks (tf.keras.layers.Conv2D) with a max pooling layer (tf.keras.layers.MaxPooling2D) in each of them.

Afterwards there is a layer to flatten the input dimension(tf.keras.layers.Flatten) before the Dense layer is added (tf.keras.layers.Dense) with 256 units. This layer is activated by a ReLU activation function (*activation='relu'*).

The neuron of a dense layer in a model receives output from every neuron of its preceding layer, where neurons of the dense layer perform matrix-vector multiplication. The keras dense layer can take a set of hyperparameters of which we used the following:

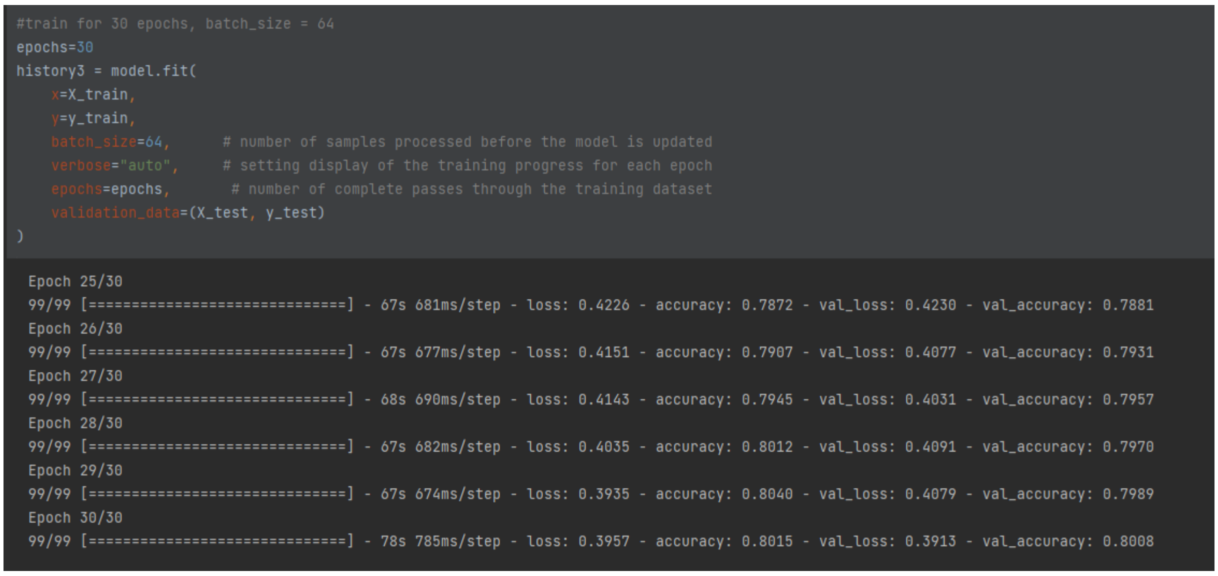
- Units: defines the size of the output from the dense layer = dimensionality of the output vector
- Activation: activation function used for the transformation of the input values of neurons -> introduces non-linearity into neural network (necessary for learning the relationship between input and output values)

The last Dense layer of the model must have the number of classes as unit parameter, which is 2 in the case of gender classification.

Between the two Dense layers we added a layer that prevents overfitting the model:

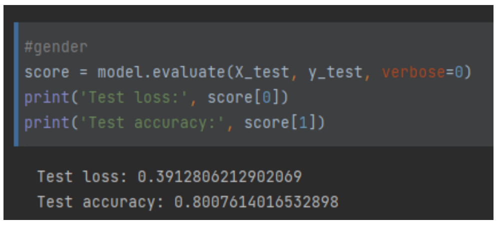
```
layers.Dropout(0.2)
```

Fitting the model was performed with 30 epochs and provided a test accuracy about 0.801.



```
#train for 30 epochs, batch_size = 64
epochs=30
history3 = model.fit(
    x=X_train,
    y=y_train,
    batch_size=64,      # number of samples processed before the model is updated
    verbose="auto",     # setting display of the training progress for each epoch
    epochs=epochs,       # number of complete passes through the training dataset
    validation_data=(X_test, y_test)
)
```

```
Epoch 25/30
99/99 [==============================] - 67s 681ms/step - loss: 0.4226 - accuracy: 0.7872 - val_loss: 0.4230 - val_accuracy: 0.7881
Epoch 26/30
99/99 [==============================] - 67s 677ms/step - loss: 0.4151 - accuracy: 0.7907 - val_loss: 0.4077 - val_accuracy: 0.7931
Epoch 27/30
99/99 [==============================] - 68s 690ms/step - loss: 0.4143 - accuracy: 0.7945 - val_loss: 0.4031 - val_accuracy: 0.7957
Epoch 28/30
99/99 [==============================] - 67s 682ms/step - loss: 0.4035 - accuracy: 0.8012 - val_loss: 0.4091 - val_accuracy: 0.7970
Epoch 29/30
99/99 [==============================] - 67s 674ms/step - loss: 0.3935 - accuracy: 0.8040 - val_loss: 0.4079 - val_accuracy: 0.7989
Epoch 30/30
99/99 [==============================] - 78s 785ms/step - loss: 0.3957 - accuracy: 0.8015 - val_loss: 0.3913 - val_accuracy: 0.8008
```



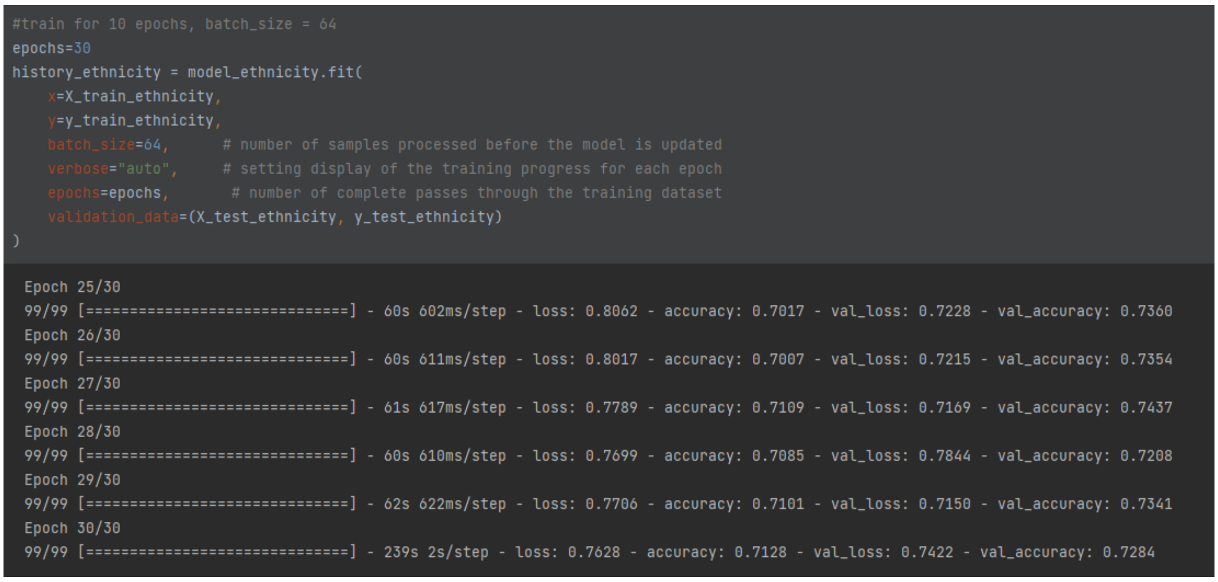
```
#gender
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.3912806212902069
Test accuracy: 0.8007614016532898
```

Ethnicity Classification

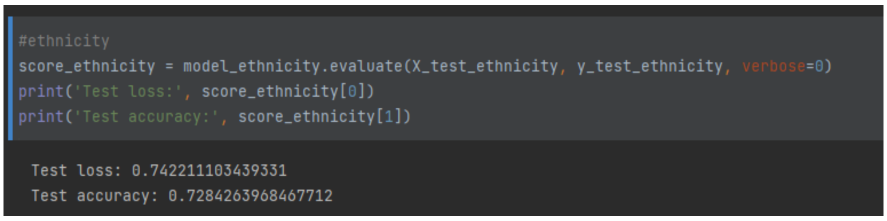
The model for Ethnicity classification is build similar to the gender classification model. Only the last Dense layer has another hyperparameter value since there are 5 classes for ethnicity.

The training was again performed for 30 epochs and resulted in a test accuracy about 0.7284.



```
#train for 10 epochs, batch_size = 64
epochs=30
history_ethnicity = model_ethnicity.fit(
    x=X_train_ethnicity,
    y=y_train_ethnicity,
    batch_size=64,      # number of samples processed before the model is updated
    verbose="auto",     # setting display of the training progress for each epoch
    epochs=epochs,       # number of complete passes through the training dataset
    validation_data=(X_test_ethnicity, y_test_ethnicity)
)
```

```
Epoch 25/30
99/99 [==============================] - 60s 602ms/step - loss: 0.8062 - accuracy: 0.7017 - val_loss: 0.7228 - val_accuracy: 0.7360
Epoch 26/30
99/99 [==============================] - 60s 611ms/step - loss: 0.8017 - accuracy: 0.7007 - val_loss: 0.7215 - val_accuracy: 0.7354
Epoch 27/30
99/99 [==============================] - 61s 617ms/step - loss: 0.7789 - accuracy: 0.7109 - val_loss: 0.7169 - val_accuracy: 0.7437
Epoch 28/30
99/99 [==============================] - 60s 610ms/step - loss: 0.7699 - accuracy: 0.7085 - val_loss: 0.7844 - val_accuracy: 0.7208
Epoch 29/30
99/99 [==============================] - 62s 622ms/step - loss: 0.7706 - accuracy: 0.7101 - val_loss: 0.7150 - val_accuracy: 0.7341
Epoch 30/30
99/99 [==============================] - 239s 2s/step - loss: 0.7628 - accuracy: 0.7128 - val_loss: 0.7422 - val_accuracy: 0.7284
```



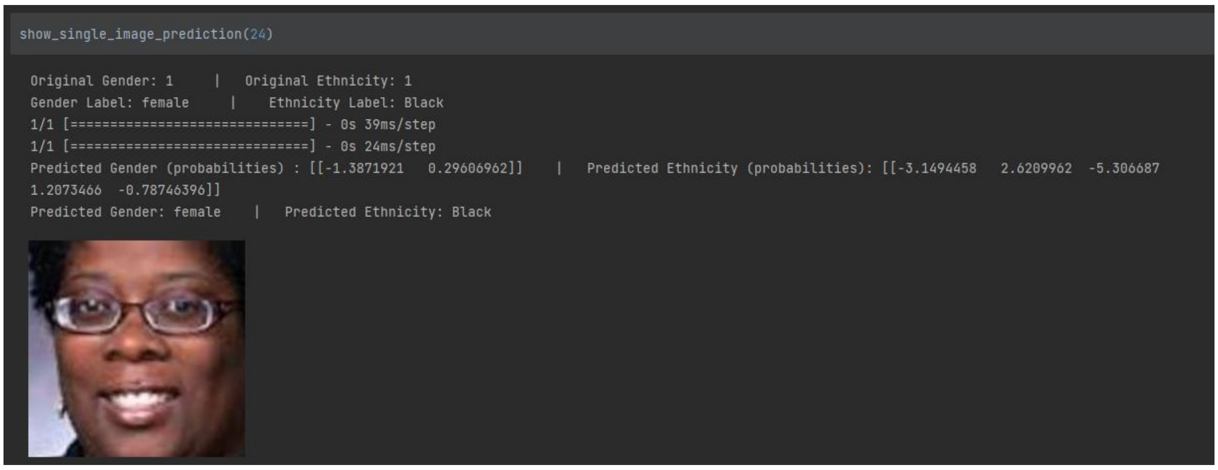
```
#ethnicity
score_ethnicity = model_ethnicity.evaluate(X_test_ethnicity, y_test_ethnicity, verbose=0)
print('Test loss:', score_ethnicity[0])
print('Test accuracy:', score_ethnicity[1])
```

```
Test loss: 0.742211103439331
Test accuracy: 0.7284263968467712
```

Results

We created plots of the accuracy and loss history for both models and defined a function for predicting the features of a single image.

The function “show_single_image_prediction(image_index)” shows the original gender and ethnicity, computes and prints the prediction, and displays the image.



```
show_single_image_prediction(24)
```

```
Original Gender: 1     |   Original Ethnicity: 1
Gender Label: female     |    Ethnicity Label: Black
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 24ms/step
Predicted Gender (probabilities) : [[-1.3871921   0.29606962]]    |   Predicted Ethnicity (probabilities): [[-3.1494458   2.6209962  -5.306687
1.2073466  -0.78746396]]
Predicted Gender: female    |   Predicted Ethnicity: Black
```

In order to print the names of the class, argmax is applied to the predicted probabilities:

```
gender_class = pred_gender_prob.argmax(axis=-1)
# sequential model returns probability for each class -> argmax for
fetching class label
```