T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

AGE PREDICTION FROM FACE USING DEEP LEARNING MODEL

SÜLEYMAN GÖLBOL

SUPERVISOR
ASSISTANT PROF. DR. BURCU YILMAZ

GEBZE
2023

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

AGE PREDICTION FROM FACE USING DEEP LEARNING MODEL

SÜLEYMAN GÖLBOL

SUPERVISOR
ASSISTANT PROF. DR. BURCU YILMAZ

2023
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 18/01/2023 by the following jury.

JURY

Member (Supervisor)	:	ASSISTANT PROF. DR. BURCU YILMAZ
Member	:	ASSOCIATE PROF. DR. HABİL KALKAN

ABSTRACT

One of the main problems of today is selling tobacco products / alcohol to underaged people in marketplaces and other stores. One of the possible solutions for this problem is checking age of the suspected person. The project report contains the details of Age Prediction From Face Using Deep Learning Model that can help to prevent these kind of issues. Solving this problem by creating a deep learning model and using transfer learning technique to improve a pretrained model are the methods that are used. A specific android mobile application which written in Flutter framework is created to connect application to models which are stored in the Azure App Services. This way, developer can provide preferable models to user without making them have to update the application. The models use UTKFace dataset and IMDB-WIKI dataset which contains over 20,000 images which are labelled by age. Created model use Convolutional Neural Network. Evaluation results contain mean absolute error as an evaluation metric. It achieves mean absolute error value below 10 laying important groundwork towards face prediction system.

Keywords: Deep Learning, Convolutional Neural Network

ÖZET

Reşit olmayan bireylere marketlerde tütün ürünleri / alkol satışı günümüzün ana problemlerinden biridir. Bu problem için olası çözümlerden bir tanesi kuşku duyulan kişinin yaşının kontrol edilmesidir. Bu ve benzeri problemleri çözmek için derin öğrenme modeli oluşturma ve önceden eğitilmiş modelleri Öğrenme Aktarımı yöntemiyle geliştirmek kullanılan yöntemlerdir. Flutter çerçevesi ile yazılmış Android mobil uygulaması, uygulamayı Azure Uygulama Servisleri üstünde saklanan modellerle bağlamak için oluşturulmuştur. Bu sayede geliştirici, kullanıcıların uygulamayı güncellemesine gerek kalmadan onlara daha gelişmiş modeller sunabilir. Modeller, 20,000'in üzerinde görüntü içeren UTKFace ve IMDB-WIKI verisetlerini kullanmaktadır. Oluşturulan model evrişimsel sinir ağlarını kullanmaktadır. Değerlendirme sonuçları ortalama mutlak hatayı değerlendirme metriği olarak içermektedir. Modelin test setindeki ortalama mutlak hata değeri 10'un altındadır ve bu yüz tahmini sistemine yönelik önemli zemin hazırlamaktadır.

Anahtar Kelimeler: Derin Öğrenme, Evrişimli Sinir Ağları.

ACKNOWLEDGEMENT

I would like to express my gratitude to my supervisor, Assistant Professor Dr. Burcu YILMAZ, for the contribution provided to this project and for guiding me through weekly meetings as well as to Associate Professor Dr. Habil KALKAN for guiding me during presentations.
Additionally, during my education, I would like to express my thanks to my family who provided me with support and knowledge.

Süleyman Gölbol

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Latest developments in Deep Learning technology has resulted in an improving way for our life. Age Recognition helps to predict the age of the person by using features of the face with feature extraction. 1.1.

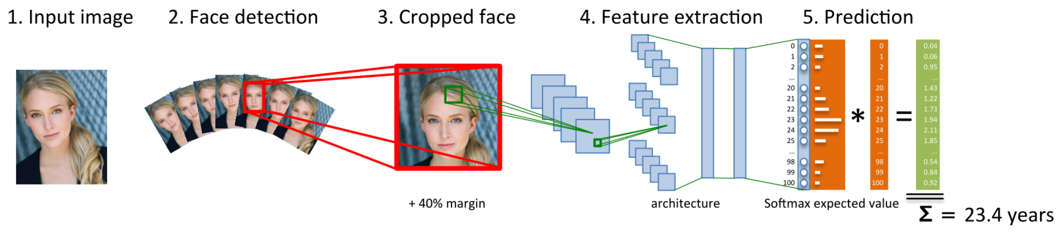

Figure 1.1: Example feature extraction figure.[1]

By using deep learning architecture needed for feature extraction, we can build a good prediction model for our purpose. Choosing the right techniques and algorithms to use in my model can be a challenging decision, and will depend on a variety of factors such as the amount of and quality of data available and the resources such as computational power and time. In general, deep learning approaches like convolutional Neural Networks (CNNs) and Residual Neural Networks (ResNet) can be more powerful for more complex tasks.

1.0.1. Convolutional Neural Network

Convolutional neural networks (CNNs) [2] are a type of neural network that are particularly well-suited for image classification and recognition tasks. They are designed to process data with a grid-like topology, such as an image, and are composed of multiple interconnected layers of artificial neurons.

The key building block of a CNN is the convolutional layer, which applies a set of learnable filters to the input data to extract features. The output of the convolutional layer is typically fed through a non-linear activation function, such as ReLU(Rectified Linear Units) which converts negative inputs to 0, before being passed on to the next layer.

CNNs also often include pooling layers, which down-sample the input data and help to reduce the size and complexity of the model. This can make the model more efficient and easier to train.

CNNs have been very successful in a wide range of applications, including image classification, object detection, and segmentation. They have been used to achieve state-of-the-art performance on many benchmarks and real-world tasks.

1.0.2. Residual Neural Network

ResNet (short for Residual Network) [3] is a type of convolutional neural network that was developed in 2015. It is designed to enable the training of very deep CNNs by addressing the problem of vanishing gradients, which occurs when the gradients of the parameters with respect to the loss function become very small and the model becomes difficult to train.

ResNet addresses this problem by using a residual learning framework, in which the input to each layer is added to the output of the previous layer, bypassing one or more layers. This allows the model to learn the residuals of the output with respect to the input, rather than the output itself.

ResNet has been very successful in a wide range of applications, including image classification, object detection, and segmentation. It has been used to achieve state-of-the-art performance on many benchmarks and real-world tasks.

WideResNet (Wide Residual Network) is a type of convolutional neural network architecture that was introduced in a 2016 paper by Sergey Zagoruyko and Nikos Komodakis. The architecture is similar to ResNet, but it uses wider layers (more filters) to increase the model's capacity and improve its performance on image classification tasks.

1.1. Face Recognition

Haar cascades [4] are a machine learning object detection method used to identify objects like faces in images. They were introduced in 2001 paper ”Rapid Object Detection using a Boosted Cascade of Simple Features”, and have been widely used in object detection tasks such as face detection, pedestrian detection, and car detection.

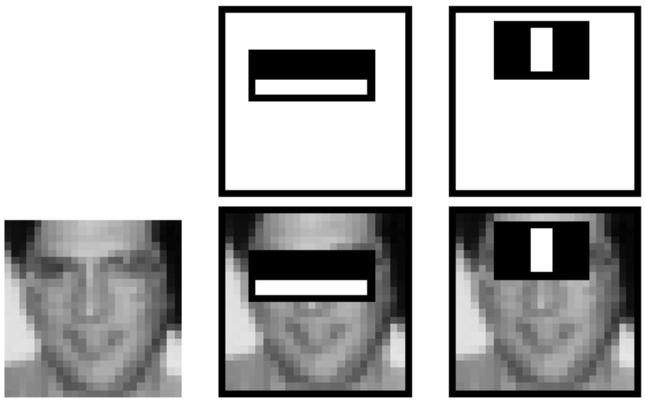
Figure 1.2: Haar Cascade Features

Haar cascades work by training a classifier on positive and negative examples of the face. The classifier is trained to identify Haar-like features in the image, which are simple differences between the sums of pixel intensities in adjacent rectangular regions. The classifier uses these features to make a decision about whether an object is present in the image or not.

Once the classifier is trained, it can be used to detect faces in new images by sliding a window across the image and applying the classifier at each location. The classifier will output a ”score” indicating the likelihood that the face is present in the window. The window is then moved to the next location and the process is repeated until the entire image has been scanned.

They are also sensitive to the size and orientation of the face, and may not work well for faces that are not well-aligned with the horizontal or vertical axes.

1.2. Age Recognition

As previously mentioned, neural networks can help for recognition tasks. Neural networks, and particularly convolutional neural networks (CNNs), have been widely used for age recognition tasks. Age recognition is the process of identifying the age of a person from an image or video of their face.

CNNs are well-suited for age recognition tasks because they are able to automatically learn complex features from the input data, and can handle large amounts of variation in the appearance of the face due to factors such as lighting, pose, and expression.

There are a number of approaches that have been proposed for using CNNs for age recognition. One approach is to directly predict the age of the person from the image using a regression model. Another approach is to classify the age into predefined age bins, such as ”child”, ”young adult”, ”middle-aged”, and ”old” or ”1-10”, ”11-20” etc.

To train a CNN for age recognition, a large dataset of images of faces with annotated ages is required. The CNN is then trained to predict the age of the person in each image. Once the CNN is trained, it can be used to predict the age of a person in a new image by inputting the image into the network and outputting the predicted age.

There are a number of challenges that need to be addressed when using CNNs for age recognition, including dealing with the large variations in the appearance of the face due to aging, and handling the large number of possible ages (e.g., ages can range from 0 to over 100).

1.2.1. Dataset Handling

To handle a dataset for age recognition tasks, there are some important keypoints I considered.

Collecting a large and diverse dataset: It is important to collect a large dataset of images of faces with labelled ages in order to train the network. The dataset should be diverse in terms of the ages of the people, the ethnicities and genders of the people, and the lighting and background conditions in the images.

Preprocessing: This includes resizing the images, converting pixel values to array values, reshaping pixel arrays, and normalizing the pixel values.

Splitting the dataset into training and test sets: The dataset should be split into a training set and a test set, with the majority of the data going into the training set and a smaller portion reserved for testing. The training set is used to train the network, while the test set is used to evaluate the performance of the trained network.

Organize the dataset: The dataset should be organized in a way that makes it easy to access and use for training and testing.

1.2.1.1. Details About Dataset

The UTKFace[5] dataset includes data information of facial images that are labeled on the basis of age, gender, and ethnicity. It includes information of more than 20000 people.

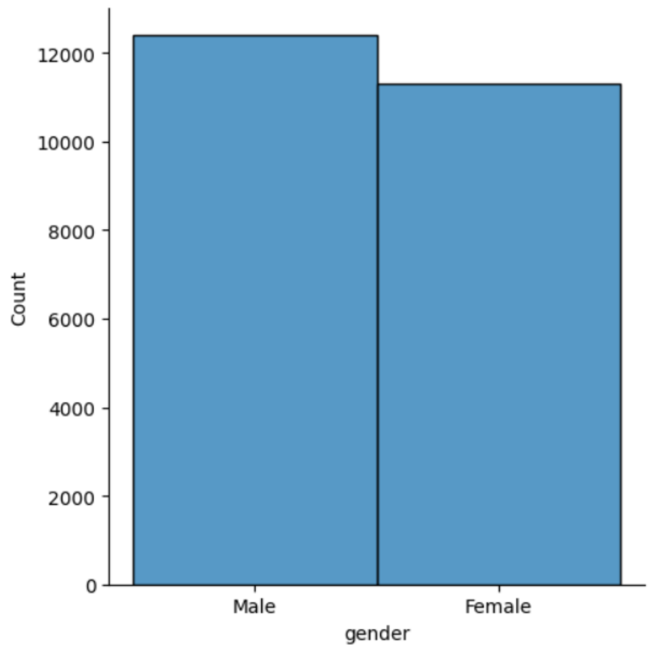

Figure 1.3: Gender Distribution

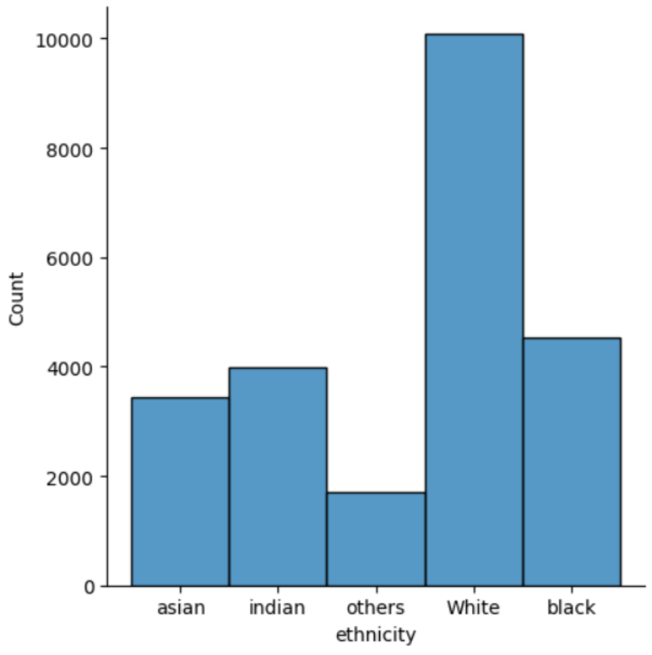

Figure 1.4: Ethnicity Distribution

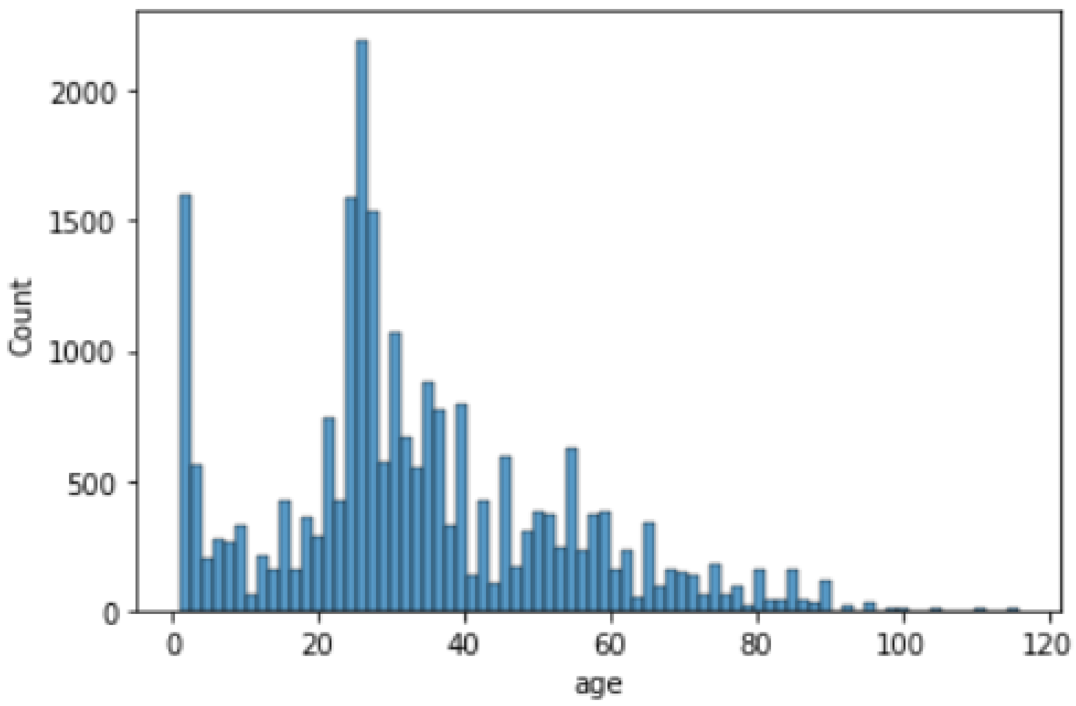

Figure 1.5: Age Distribution

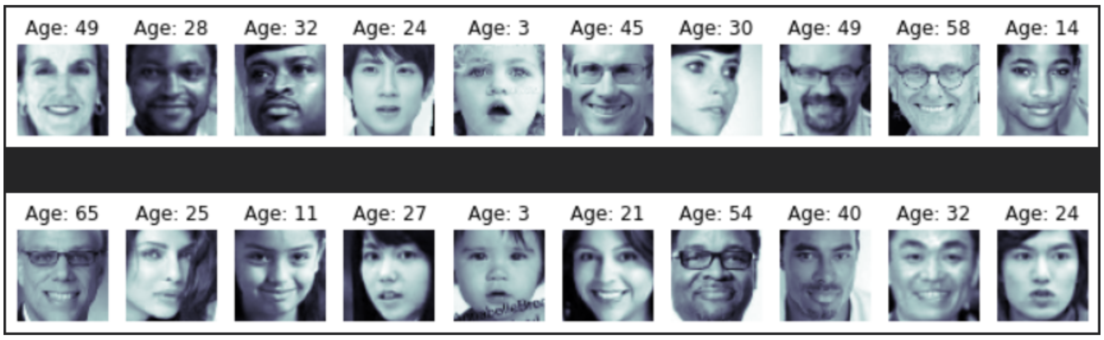

Figure 1.6: Example images from dataset

1.3. Mobile Application

Android Mobile application is created by using Dart programming language with Flutter framework.

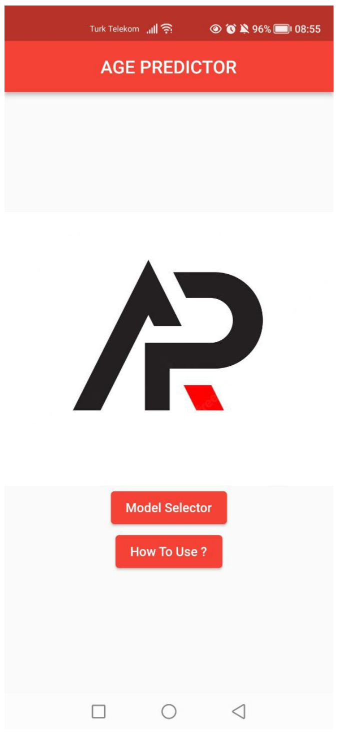

Figure 1.7: Main Page of Mobile Application

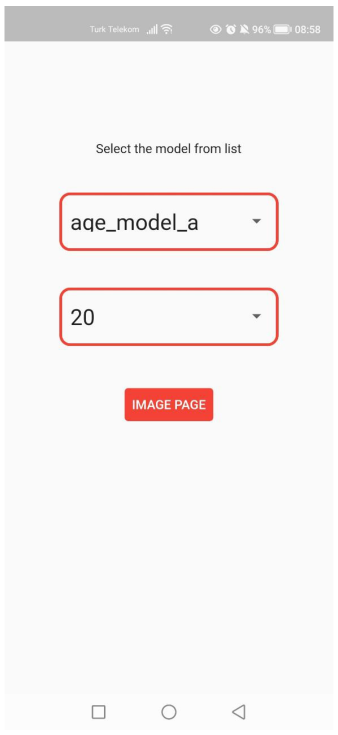

Figure 1.8: Model Selector Page

Model selector page allows users to select different models with different epochs.

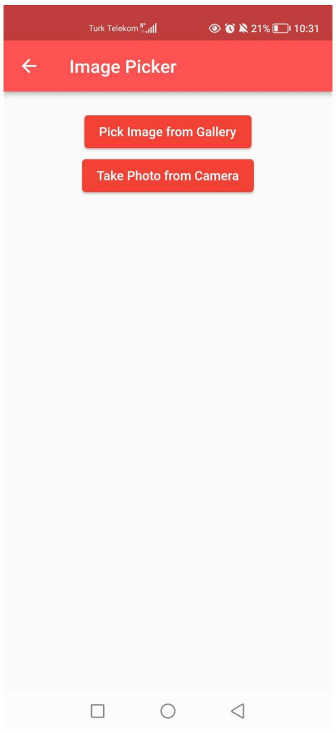

Figure 1.9: Image Picker Page

For selecting image from gallery and taking a picture; Flutter Image Picker package is used. The image is converted to base64 format when sending the image.

1.3.0.1. Base64

Base64 encoding is a way to represent binary data, such as an image, as a string of ASCII characters. This can be useful when the binary data needs to be transmitted over a network or stored in a text-based format, such as JSON.

To encode an image in Base64, the image is first converted to a binary representation, such as a byte array. The binary data is then passed through a Base64 encoding function, which produces a string of ASCII characters representing the binary data.

To decode a Base64-encoded image, the string of ASCII characters is passed through a Base64 decoding function, which converts the string back into the original binary data. The binary data can then be used to reconstruct the original image.

In JSON, a Base64-encoded image can be stored as a string value in a JSON object. For example, the following JSON object stores an image as a Base64-encoded string:

```
{
"image": "iVBORw0KGg....."
}
```

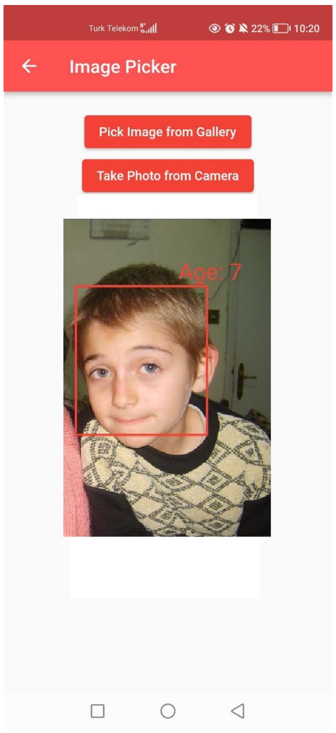

Figure 1.10: Example output after connection and prediction

1.4. Azure App Services

Azure App Services is a cloud platform service provided by Microsoft Azure that allows developers like me to build, deploy, and host web, mobile, and API applications. It provides a range of features and tools to help developers build, test, and manage applications, including support for a variety of programming languages, integrations with popular development tools, and automatic scaling and load balancing.

One of the key features of Azure App Services is the ability to deploy machine learning models as web services. This allows developers to deploy trained models as APIs that can be accessed over the web, enabling other applications or users to make predictions or inferences using the model.

To deploy a machine learning model as a web service using Azure App Services, I used function app service, which provides tools and libraries for building and deploying machine learning models. Azure App Services allow me to deploy trained models as web services and manage the APIs through the Azure portal.

1.5. Automated Machine Learning Deployment

Automated machine learning deployment with GitHub Actions refers to the process of automatically deploying a machine learning model as a web service or API using GitHub Actions. GitHub Actions is a continuous integration and delivery (CI/CD) platform that is built into GitHub and allows to automate software development workflows.

To deploy a machine learning model using GitHub Actions, I have created a workflow that is triggered by a specific event, such as updating the repository or change in a model. The workflow can then run a series of tasks, such as packaging it for deployment, and deploying it to a hosting platform.

I used Azure Machine Learning to automate machine learning deployment with GitHub Actions. It is integrated into GitHub Actions workflows through the use of action plugins or custom scripts.

2. IMPLEMENTATIONS

Implementation Chapter contains the implementation details of the project.

2.1. Splitting Dataset

To split the dataset, I created a function to randomly split dataset in a percentage of 70/30.

```
  def shuffle_split_data(X, y):
    arr_rand = np.random.rand(X.shape[0])
    split = arr_rand < np.percentile(arr_rand, 70)

    X_train = X[split]
    y_train = y[split]
    X_test =  X[~split]
    y_test = y[split]

    # print (len(X_train), len(y_train), len(X_test), len(y_test))
    return X_train, y_train, X_test, y_test

train_x, train_y, test_x, test_y = shuffle_split_data(pix_vals, age.values)
```

2.2. Creating Model

To create model, I created model as Sequential which is a type of model that is composed of a linear stack of layers, where the output of one layer is used as the input to the next layer.

```
model = keras.Sequential()
```

Then, I added 3 convolutional layers with 64, 128, and 256 filters which is a common design pattern in convolutional neural networks (CNNs). These layers are followed by pooling layers, which down-sample the input data, and fully-connected (dense) layers, which perform classification or regression on the features extracted by the convolutional layers.

There are a few reasons why adding 3 convolutional layers with increasing numbers of filters can be a good design choice for a CNN:

- Feature extraction: [6] Convolutional layers are able to extract features from the input data by applying a set of learnable filters to the input. Adding more convolutional layers with more filters can allow the model to learn more complex and nuanced features from the input data.
- Invariance: Convolutional layers are designed to be translation invariant, meaning that they are able to recognize the same features regardless of their position in the input data. Adding more convolutional layers can help the model to learn more complex features that are invariant to translation, which can be useful for tasks like this.
- Capacity: Adding more convolutional layers with more filters increases the capacity of the model, allowing it to learn more complex patterns in the data. But it is also important to ensure that the model is not over-capacity, as this can lead to overfitting and poor generalization to new data.

```
model.add(Conv2D(64, kernel_size=(3,3),
input_shape=pix_vals.shape[1:], activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())

model.add(Conv2D(128, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(Conv2D(256, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(1))
```

After these, I compiled the model with mean squared error loss function and mean absolute error as metric:

```
model.compile(optimizer='adam',
          loss='mean_squared_error',
          metrics=['mae'])
```

The loss parameter in the model compilation step specifies the loss function that will be used to evaluate the model during training. The mean_squared_error loss function is a common choice for regression tasks, and it calculates the mean squared difference between the predicted values and the true values.

The metrics parameter specifies a list of metrics that will be used to evaluate the model. In this case, the mae metric stands for mean absolute error, which is another measure of error used in regression tasks. It is calculated as the mean absolute difference between the predicted values and the true values.

Table 2.1: Error table.

Prediction	Actual	Absolute Error	Squared Error
140	120	20	400

By including both the mean_squared_error loss function and the mae metric in the model compilation step, it specifies using the mean squared error loss function to guide the training process, and to report the mean absolute error as an evaluation metric. This is useful for getting a sense of both the average magnitude of the error and the average squared error in the model's predictions.

2.3. Mobile App Azure Connection

For connection of application to Azure, I used several methods.

```
Future<String> convertFileToImage(File picture) async {
  File compressedPicture = await FlutterNativeImage.compressImage(
            storedImageOnRoot.path, quality: 50, percentage: 80);
  List<int> imageBytes = compressedPicture.readAsBytesSync();

  String base64Image = base64.encode(imageBytes);
  return base64Image;
}
```

This method compresses the image before encoding with base64 to use as string in json data.

```
Future<String> trialPost() async {
  // convert image to json using base64 encoding
  var imageJson = await convertFileToImage(storedImageOnRoot);
  var jsonCode = {"model_path": modelsPath + model, "image": imageJson};
```

```
    var body = jsonEncode(jsonCode);

    var response =
        await http.post(Uri.https(
        'agepredictorwin2.azurewebsites.net', '/api/HttpTrigger1'),
        body: body, headers: {"x-functions-key": secretKey});
  }
```

This method posts a http request to function app on web using HtppTrigger function with API key.

```
  def face_border_detector(self, image):
        face_cascade = cv.CascadeClassifier(
        'haarcascade_frontalface_default.xml')
        gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.1, 4)
        biggest_face = [0,0,0,0]
        for i, (x, y, w, h) in enumerate(faces):
            if (biggest_face[2] - biggest_face[0]) < (w - x):
                biggest_face = [x, y, w, h]

        [x,y,w,h] = biggest_face
        cv.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)
        return biggest_face
```

This method detects the border of face using haar cascades method. It finds the biggest face in image and returns the coordinates.

```
  def main(req: func.HttpRequest) -> func.HttpResponse:
    headers = {
        "Content-type": "application/json",
        "Access-Control-Allow-Origin": "*"
    }

    model_path = req.params.get('model_path')
    image = req.params.get('image')

    req_body = req.get_json()
    model_path = req_body.get('model_path')
    image = req_body.get('image')
    model = InferenceModel(model_path)
    preds = model.predict(image)
    results = json.dumps({"preds": preds}, default=int32_to_int)
    return func.HttpResponse(json.dumps(results), headers=headers)
```

This is the simplified main method in htpp request function. When it gets the request as json file, it loads the model, converts base64 string to image. It calls predict function so that it returns the age predicted.

3. EVALUATION

Evaluation Chapter contains the evaluation details of the project.

3.1. Fitting the Model

Fitting the model with epoch value more than 30 results like this:

```
Epoch 1/38
260/260 [==============================] - 4m 14ms/step - loss: 209.5192
- mae: 10.7432 - val_loss: 1909.9983 - val_mae: 40.0445
Epoch 2/38
260/260 [==============================] - 3m 13ms/step - loss: 130.4871
- mae: 8.5323 - val_loss: 1218.8787 - val_mae: 32.0797
Epoch 3/38
.
.
Epoch 36/38
260/260 [==============================] - 4m 14ms/step - loss: 38.6927
- mae: 4.6639 - val_loss: 143.1067 - val_mae: 9.1923
Epoch 37/38
260/260 [==============================] - 4m 14ms/step - loss: 38.9278
- mae: 4.6550 - val_loss: 78.9507 - val_mae: 6.5396
Epoch 38/38
260/260 [==============================] - 4m 14ms/step - loss: 38.7480
- mae: 4.6669 - val_loss: 104.8647 - val_mae: 7.1492
```

3.2. Evaluating

```
mse_loss, mae = model.evaluate(test_x, test_y, verbose=1)
```

After evaluating, these are the results:

```
    223/223 [==============================] - 1s 4ms/step - mse_loss:
104.8647 - mae: 7.1492
```

As expected, validation results of mean absolute error (val_mae) is bigger than mean absolute error. Validation mean absolute error (MAE) is higher than the training

MAE when fitting a machine learning model. This is because the validation MAE is an estimate of the generalization error of the model, which is the error that the model would make on new, unseen data. The training MAE, on the other hand, is an estimate of the error made by the model on the training data, which the model has seen during training.

It is expected that the model will perform better on the training data than on new, unseen data, as the model has been specifically designed to fit the training data.

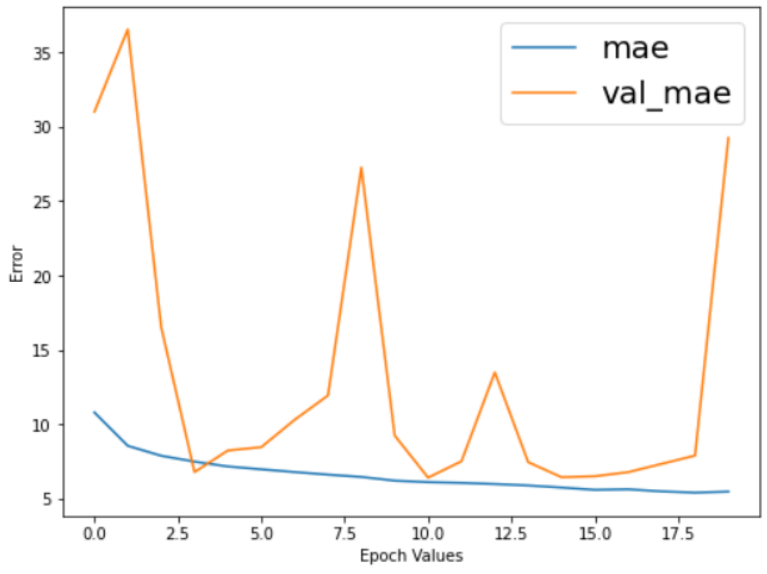

Figure 3.1: Output of mae results for 20 epochs

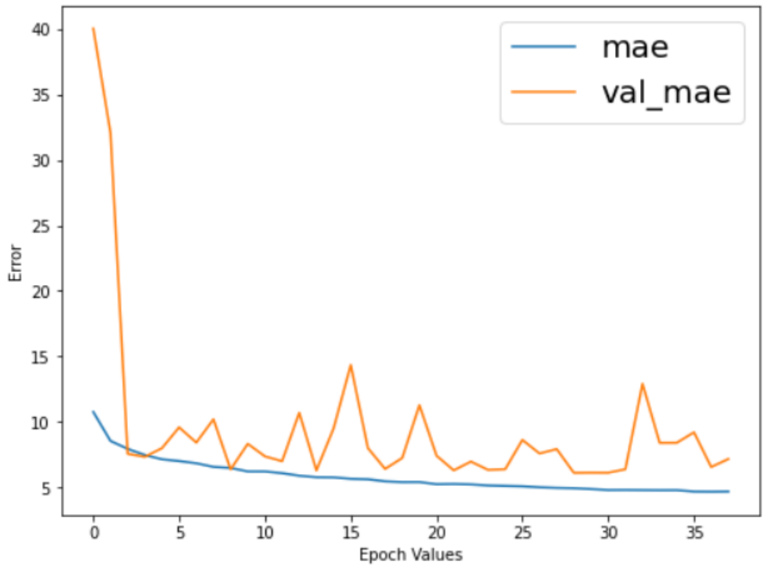

Figure 3.2: Output of mae results for 38 epochs

Table 3.1: Comparison of errors.

Model	Epochs	MSE VAL LOSS	MAE
My Model	20	1043.81	29.25
My Model	38	104.86	7.14

Unfortunately, I couldn't finish transfer learning the early pre-trained model since I ran into problems while unfreezing.

4. CONCLUSIONS

I attempted to develop a age prediction system using deep learning model with mean squared error loss and mean absolute error metric. The first phase is to create a mobile application that selects models and images with converting them into proper data format for communication. Second step is to create/transfer learn the model with proper architectures for better results. Third phase is to deploy the model as endpoint artifact in Azure Function App. The experiment results indicate that it’s possible to create a model with small errors.

BIBLIOGRAPHY

[1] G. D. Greenwade, “The Comprehensive Tex Archive Network (CTAN),” *TUGBoat*, vol. 14, no. 3, pp. 342–351, 1993.

[2] R. Chauhan, K. Ghanshala, and R. Joshi, “Convolutional neural network (cnn) for image detection and recognition,” pp. 278–282, Dec. 2018. DOI: `10.1109/ICSCCC.2018.8703316`.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015. DOI: `10.48550/ARXIV.1512.03385`. [Online]. Available: `https://arxiv.org/abs/1512.03385`.

[4] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” vol. 1, 2001. DOI: `10.1109/CVPR.2001.990517`.

[5] Z. Zhifei and Y. Hairong, “Age progression/regression by conditional adversarial autoencoder,” 2017.

[6] S. Dara and P. Tumma, “Feature extraction by using deep learning: A survey,” pp. 1795–1801, 2018. DOI: `10.1109/ICECA.2018.8474912`.