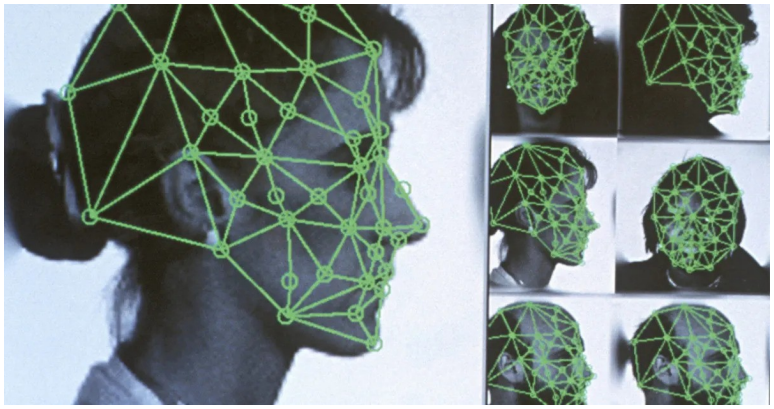
Machine Learning 2

Coursework 1: Age Estimation and Gender Classification

University of Bath



ID	Contribution	Agree? [Y/N]
oir20	50%	Y
vs929	50%	Y

No.	Model Links
1.	The custom CNN model
2.	The pre-trained CNN model

Number Pages: 10
Word Count: 1495

Table of Contents

List of Figures

Coursework Report

1.1 Introduction

The purpose of this assignment is to use a subset of the UTKFace dataset to build and train Convolutional Neural Networks (CNNs) for age estimation and gender classification. CNNs outperform traditional models in age and gender classification, handling facial variations more effectively [1]. Humans are quite accurate at guessing a person's age and gender. However, due to differences in facial characteristics, and image quality, training a model to perform similarly is a difficult task.

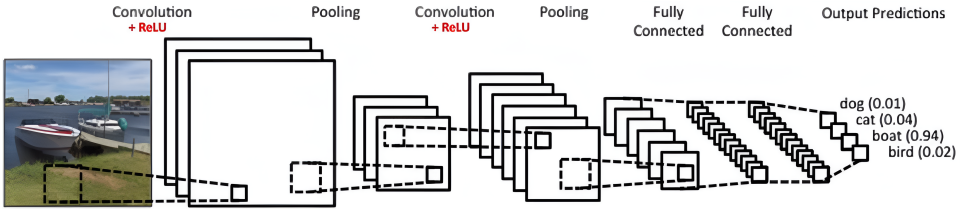


Figure 1.1: Structure of a CNN

CNNs are deep learning models that automatically learn spatial features and patterns, making them effective for computer vision tasks. As shown in Figure 1.1, they consist of multiple layers, including convolutional layers, aggregation layers, and fully connected layers, which all work together to extract and learn features from input data [2].

The images come from the UTKFace dataset, which contains over 20,000 faces labeled with age, gender, and ethnicity. As seen from Figure 1.2, the images have a wide age range (0–116 years) and varied image quality, angles and expressions, providing a dataset perfect for this task.

Figure 1.2: Samples from the UTKFace dataset

This assignment involves training two CNN models:

1. **A Custom CNN Model:** Designed from scratch with certain restrictions.
2. **A Fine-Tuned Pretrained Model:** A pre-existing CNN model will be used and fine-tuned on the dataset to improve its performance.

Both models were trained using Google Colab. We describe each architecture, outline the training process, and compare performance, highlighting key insights and challenges.

1.2 The custom CNN model

The architecture begins with a shared feature extractor that captures visual patterns relevant to both tasks, followed by two distinct branches, one for gender and one for age, each optimized for its specific goal. The reason we do this is that features that would be important for age are not necessarily important for gender, and vice versa. For example, wrinkles are important to understand how old a person is but do not help in classifying a person's gender.

We define the input layer and apply our data augmentation to it, consisting of horizontal flip and random rotation of 10%. This is to improve performance since our dataset is small [3]. We then build the shared layer, where there are three convolutional blocks (Figure 1.3). To capture fine spatial information, we use small filters (3×3 and 2×2). As the filter depth increases (16 -¿ 32 -¿ 64), we capture more complicated patterns. Each convolution is followed by batch normalization to stabilize training. ReLU activations introduce non-linearity, while max-pooling reduces spatial dimensions. To prevent overfitting, each convolutional layer uses L2 regularization to penalize big weights.

```
# Define model architecture
base = Sequential([

    # Convolutional layer 1
    layers.Conv2D(16, (3, 3), use_bias=False, kernel_regularizer=l2(l2_val), input_shape=input_shape, strides=1,
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D((2, 2)),

    # Convolutional layer 2
    layers.Conv2D(32, (3, 3), use_bias=False, kernel_regularizer=l2(l2_val), input_shape=input_shape, strides=1,
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D((2, 2)),

    # Convolutional layer 3
    layers.Conv2D(64, (2, 2), use_bias=False, kernel_regularizer=l2(l2_val), strides = 1, padding='valid'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D((2, 2)),
])
```

Figure 1.3: Custom CNN Shared Layer

The gender classification branch continues with a convolutional layer, with batch normalization, ReLU, and max pooling to get the filter size under 10x10. It then flattens the output and passes it through a dense layer with 64 units, followed by batch normalization, ReLU activation, and a dropout of 0.85. While this is high, we found it is required to prevent overfitting. The final dense layer has sigmoid activation, which is effective as it outputs a probability score that can be interpreted as the likelihood of an input belonging to a particular class, such as male or female [4].

The age regression branch continues with two more convolutional layers (128 and 256), also followed by batch normalization, ReLU, and max pooling. This

is justified by the higher complexity of predicting a continuous number. The output is then flattened and passed through two dense layers (512 and 128 units), each followed by an L2 regularizer, dropout, and ReLU. The final output layer uses ReLU, which outputs the input directly if it is positive, otherwise, it outputs zero. This helps in preventing the vanishing gradient problem, which is common in deep networks [4].

The model is constructed using mean absolute error (MAE) for age and binary cross-entropy for gender, which match their tasks. When validation loss no longer improves, with a patience of five and 'restore_best_weights', training is stopped early to avoid overfitting. A learning rate scheduler allows for greater steps early and finer learning later by decaying the learning rate after 10 epochs. We found that this works better for our model compared to the usual optimizers used for CNNs, like Adam [5].

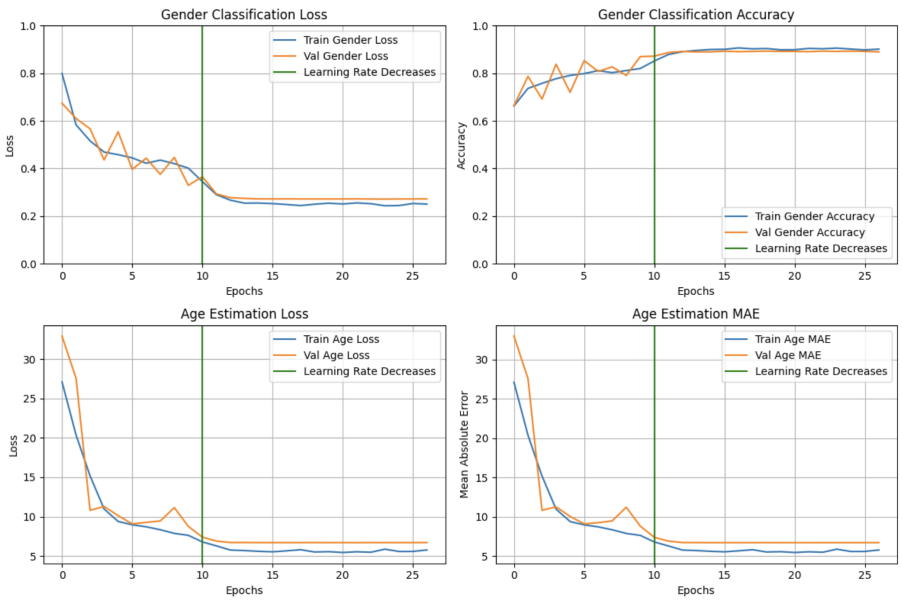


Figure 1.4: Model A Results

After running 30 epochs, the validation gender accuracy reached 90.42%, meaning the model learned to distinguish gender from facial features. The validation age estimation MAE was approximately 6.58 years, which is reasonable for such a complex task. From Figure 1.4, we see that the training curves indicate smooth convergence for both tasks. Gender loss and accuracy stabilized after epoch 10, and age MAE showed some instability early on but improved after learning rate reduction.

1.3 The pre-trained CNN model

We use a pretrained VGG16 model as the base of our architecture. VGG16 is a widely used CNN known for its depth and performance, with 13 convolutional and 3 fully connected layers [6]. We used VGG16 to leverage its strong feature extraction, especially for our small dataset. Its early layers capture general patterns like edges and textures, providing a solid foundation for face-related tasks.

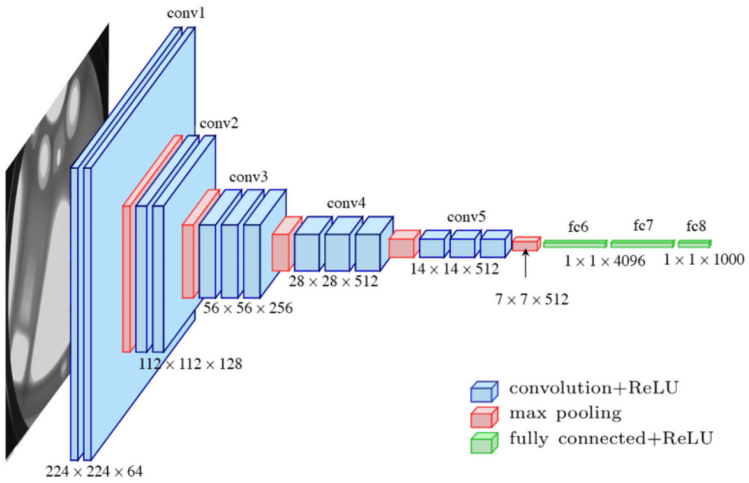


Figure 1.5: VGG16 Architecture

We loaded VGG16, with weights set to 'imagenet', without its top classification layers and froze all of its layers except the last three. This allowed the network to keep its pre-trained knowledge while still adapting to our tasks. The input was passed through the same data augmentation layer and then fed into the frozen VGG16 base with inference mode to prevent updates during backpropagation for most of its layers.

The output from VGG16 was then flattened and split into two separate branches, like our previous model. The gender branch consisted of two dense layers with 128 and 256 units, both followed by L2 regularizer and dropout layers (0.3 and 0.5) to reduce overfitting. The final gender output also uses a sigmoid activation.

The age branch also began with dense layers, increasing from 256 to 512 units. These layers were followed by batch normalization and ReLU activations. A moderate dropout rate of 0.2 was used after each dense layer to prevent overfitting while keeping useful signals. The final output used a ReLU activation to again ensure non-negative predictions for age.

The model was compiled using the Adam optimizer, with a learning rate of 1e-4, which offered a good balance between fast convergence and training stability. We used Adam here as our scheduler did not perform as well as Adam. Binary cross-entropy was again used as the loss function for gender classification, while MAE was used for age regression. Early stopping was used with the same conditions to help avoid overfitting.

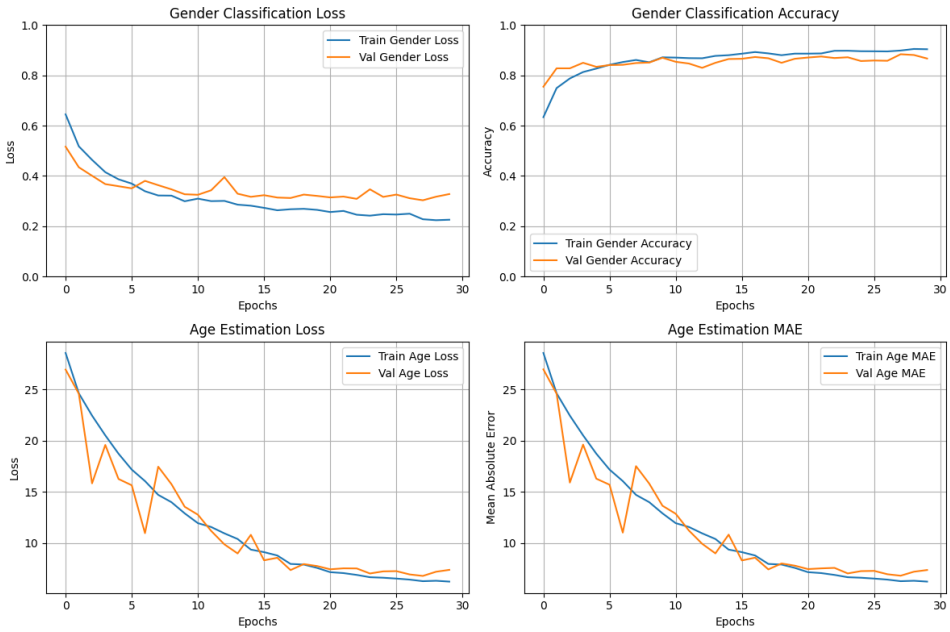


Figure 1.6: Model B Results

The performance of our transfer learning model also shows strong results. With a final gender accuracy of 90.15%, the model effectively uses VGG16's feature extraction capabilities to predict genders. The age estimation task resulted in an MAE of 6.9292, indicating an average prediction deviation of about 7 years. The training curves show a consistent decline in both training and validation loss, indicating stable learning with minimal overfitting.

1.4 Summary and discussion

Based on test results using a subset of 10,000 images from the UTKFace dataset, both the custom and pre-trained VGG16-based models show strong performance in gender classification and age estimation, with the custom model consistently outperforming the pre-trained one.

```
# Configs
DATASET_PATH = "/Users/s16teen/Downloads/Work/UOB/SEM2/ML2/age_gender_estimation_CNN/Evaluation/UTKFace"
IMAGE_SIZE = (128, 128)
SAMPLE_SIZE = 10000

# Load Filenames
all_image_files = [
    file for file in os.listdir(DATASET_PATH)
    if file.lower().endswith('.jpg')
]

# Sample a subset for evaluation
all_image_files = np.random.choice(all_image_files, SAMPLE_SIZE, replace=False).tolist()
```

```
# Load images and labels
def load_images_and_labels(dataset_path, filenames, target_size=(128, 128)):
    print("Loading image data and extracting labels...")
    images, age_labels, gender_labels = [], [], []

    for file in filenames:
        img_path = os.path.join(dataset_path, file)
        img = cv2.imread(img_path)
        img = cv2.resize(img, target_size)
        img = img / 255.0

        try:
            age, gender = file.split('_')[:2]
            age_labels.append(int(age))
            gender_labels.append(int(gender))
            images.append(img)
        except ValueError:
            continue

    return np.array(images), np.array(age_labels), np.array(gender_labels)

# Load the data
images, age_labels, gender_labels = load_images_and_labels(DATASET_PATH, all_image_files, IMAGE_SIZE)
```

```
Loading image data and extracting labels...
```

```
# Load Models
modelA = load_model('age_gender_A.keras')
modelB = load_model('age_gender_B.keras')

# Predict using both models
predA = modelA.predict(images)
predB = modelB.predict(images)
```

Figure 1.7: Evaluating the models on a custom test set

Both the custom and pre-trained VGG16-based models performed well in gender classification and age estimation. Despite using the same dataset and training procedures, the custom model showed slightly better results, indicating its task-specific design may offer a performance advantage.

1.4.1 Gender Classification

The custom model achieved a slightly higher gender classification accuracy (89.5%) compared to the pre-trained model (88.1%). This suggests the custom model is slightly better at learning gender-related features, despite both models reaching similar accuracy during validation.

1.4.2 Age Estimation

In age estimation, the custom model again performed better, with a lower MAE of 6.10 years versus 6.37 years for the pre-trained model. This confirms the custom architecture's stronger ability to extract age-relevant features, likely due to its task-specific design.

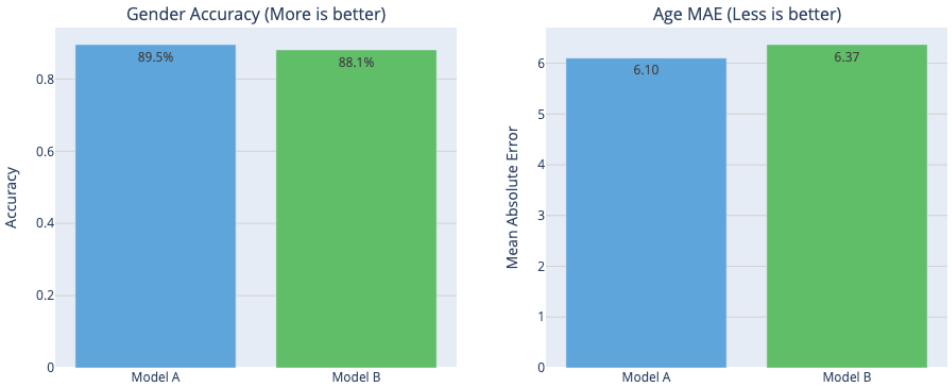


Figure 1.8: Evaluation results

1.4.3 Discussion

We tested different configurations to improve performance. We tried adjusting strides and pooling methods, finding max pooling more effective than changing strides. We experimented with filter sizes, dense layer neurons, and the use of GlobalAveragePooling2D instead of flatten. Additionally, we tried using shared convolutional layers for both age and gender before splitting, along with different regularization values. We tested different epoch numbers, batch sizes, and learning rates, as well as compared MSE and MAE loss functions. We also fine-tuned the number of frozen layers in the pre-trained VGG16 model. Through trial and error, we identified the optimal configurations for our task.

1.4.4 Applying Deep Learning to real life

CNNs have shown success in real-life applications, they are pivotal in medical imaging, helping with disease detection and diagnosis [7]. They are used in NLP for sentiment analysis and text classification [7]. In agriculture, they can enhance crop monitoring and pest detection [8]. And they enhance security systems by improving facial authentication [9]. These advancements showcase the transformative potential of deep learning with CNNs.

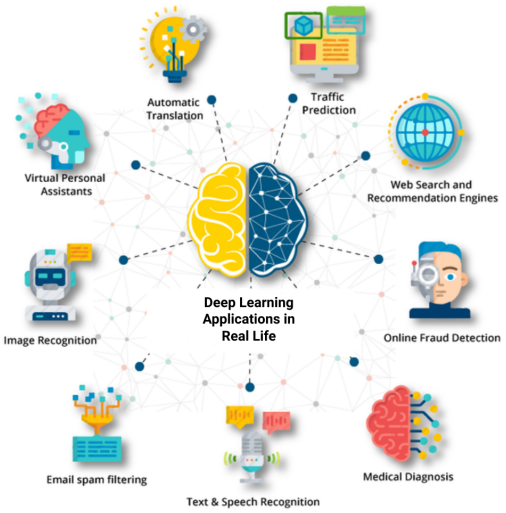


Figure 1.9: Real life applications of deep learning.

1.4.5 Limitations and Future Work

1. **Small Dataset Size:** The model was trained on a small dataset, limiting its generalization.
2. **Limited Training Time and Compute:** Training was limited by free Google Colab credits, restricting time and hardware and preventing full fine-tuning.
3. **Use More Advanced Architectures:** Try out other pretrained models like EfficientNet, ResNet50, or MobileNetV3.
4. **Hyperparameter Tuning:** With more compute, we could optimize hyperparameters like learning rates, regularization, and layer sizes to boost performance.

Bibliography

[1] Mohammed Kamel Benkaddour. Cnn based features extraction for age estimation and gender classification. *Informatica (Slovenia)*, 45, 2021. 1

[2] Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Chau. Cnn explainer: Learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27:1396–1406, 2020. 1

[3] Aleksei Zhuchkov. Analyzing the effectiveness of image augmentations for face recognition from limited data. In *2021 International Conference" Nonlinearity, Information and Robotics"(NIR)*, pages 1–6. IEEE, 2021. 3

[4] Kunal Jain, Muskan Chawla, Anupma Gadhwal, Rachna Jain, and P. Nagrath. Age and gender prediction using convolutional neural network. *Lecture Notes in Networks and Systems*, 2020. 3, 4

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[6] Shallu Sharma and R. Mehra. Breast cancer histology images classification: Training from scratch or transfer learning? *ICT Express*, 4:247–254, 2018. 5

[7] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021. 9

[8] A. Kamilaris and F. X. Prenafeta-Boldú. A review of the use of convolutional neural networks in agriculture. *The Journal of Agricultural Science*, 156(3):312–322, 2018. 9

[9] Xia Zhao, Limin Wang, Yufei Zhang, Xuming Han, Muhammet Deveci, and Milan Parmar. A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, 57(4):99, 2024. 9