ENSC 813: Deep Learning Systems in Engineering

Kumar Abhishek

Assignment 4 July 28, 2021 Student ID: 301371793

Preparing the datasets

UTKFace: Aligned&Cropped Faces dataset

There are 9,780 images in the UTKFace: Aligned&Cropped Faces (UTK hereafter) dataset. As the question says, we filter the dataset to only include images of people with ages 13 years or above. Upon applying this filtering, we are left with 6,525 images.

The dataset contains gender labels in the image name using 0 or 1 to denote male or female respectively. However, upon inspection, we found that one file (**61_3_20170109150557335.jpg.chip.jpg**) had a gender label 3, which is an invalid label. Therefore, we removed this file and we are left with 6,424 images with breakdown as shown in Table 1.

Gender	Count
Male (0)	2829
Female (1)	3695
Female percentage: 56.64%	

Table 1: Class-wise population of the UTK dataset.

Adience Benchmark Gender And Age Classification dataset

Similar to the UTK dataset above, we filter the images from the Adience Benchmark Gender And Age Classification (Adience hereafter) dataset to include only those of people 13 years or older. Interestingly, the Adience dataset does not have exact ages for all images and instead uses a mixture of exact ages and age intervals (e.g., **(4, 6)**, **(8, 12)**, etc.). To avoid any underage images being used, we have filtered any images whose upper bound of the age interval is more than 13, meaning the age interval **(8, 23)** was filtered out.

After this filtering, we were left with 11,769 images. Since this is considerably larger than the UTK dataset, we split the Adience dataset into training, validation, and testing partitions. An important thing to keep in mind when splitting is to retain the class-wise distribution for all variables of interest between the three partitions. This means that the ratio of males to females must be the same in each of the three partitions as that in the original dataset. Additionally, in order to avoid unbalanced distribution of age groups across partitions, we also ensure that all the partitions have the same proportions of various age groups. This is done by stratified splitting and we perform the stratification along 2 columns: age and gender. The breakdown of the three splits is as shown in Table 2.

Split	Gender	Count	Total (Female percentage)
train	Male (0)	3881	8242 (**52.91%**)
	Female (1)	4361	
valid	Male (0)	417	881 (**52.67%**)
	Female (1)	464	
test	Male (0)	1246	2646 (**52.91%**)
	Female (1)	1400	

Table 2: Class-wise population of the Adience dataset.

Combining the two datasets

Finally, we combine the two datasets to train and evaluate our gender classification model. As we saw above, the Adience dataset is much larger, so we merge the UTK dataset with the training partition of the Adience dataset and use this for training. For validation and testing, we use the respective partitions from the Adience dataset. Table 3 presents the detailed breakdown of the partitions used to train, validate, and test the classification models.

Split (Dataset)	Gender	Count	Total (Female percentage)
train (UTK + Adience)	Male (0)	6710	14766 (**54.56%**)
	Female (1)	8056	
valid (Adience)	Male (0)	417	881 (**52.67%**)
	Female (1)	464	
test (Adience)	Male (0)	1246	2646 (**52.91%**)
	Female (1)	1400	

Table 3: Class-wise population of the training, validation, and testing splits used.

As we can see from Table 3, the male:female ratio is very close to 1:1 and therefore the classes are almost perfectly balanced. As such, there is no need to address the problem of class imbalance.

Training gender prediction models

Logistic regression neuron

The logistic regression classifier neuron has been defined in **utils.py** as a custom class named **LogisticRegression**. This neuron takes in a 200 x 200 RGB image and predicts a single value between 0 and 1, where 0 corresponds to male and 1 corresponds to female. As such, the neuron takes in a 200 x 200 x 3 = 120000 dimensional input and has ≈120k parameters. Since the implementation is in PyTorch, we use the **BCEWithLogitsLoss()** function to train the model since it is numerically more stable than using binary cross entropy loss with sigmoid activation function[1].

In order to visualize the training process, we plot the loss values and the classification accuracy of the neuron over the course of training for all the datasets in Figure 1.

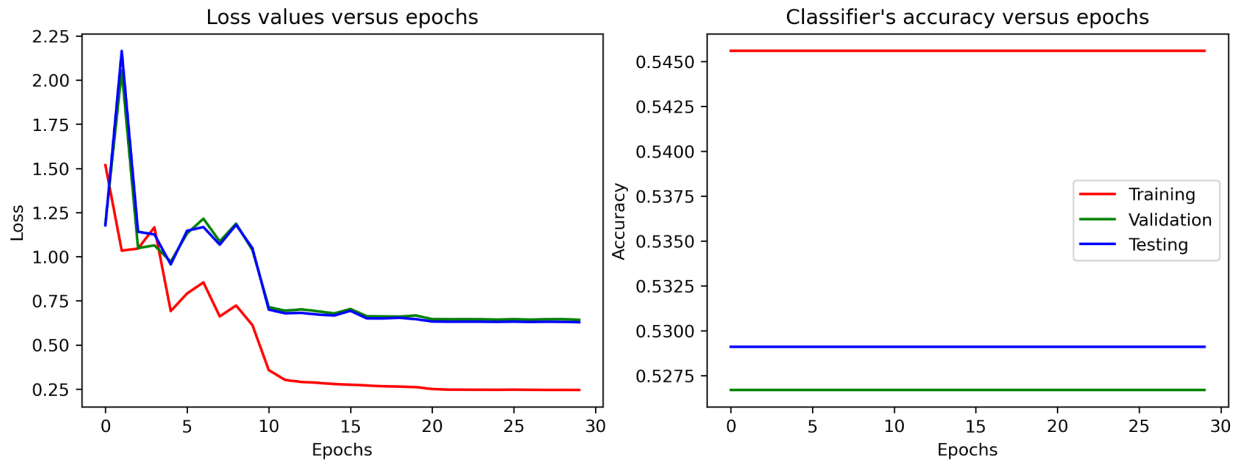

Figure 1: The classifier's loss and accuracy plots over the course of training.

As we can see in Figure 1, the loss values for all the dataset partitions converge but remain fairly large. This means that the model's capacity is not enough for this dataset. Similarly, the classifier's accuracy for all the dataset partitions is around 52%-54%, indicating that although the model is able to learn upto some extent (as also supported by the decreasing loss values), it stops learning very quickly.

Therefore, this model is not adequate for this classification task. This can be explained easily by the low capacity of this single neuron classifier. Since the images are 200 x 200 RGB, we need to draw a decision boundary in $\mathbb{R}^{120000}$. However, a logistic regression neuron, despite the sigmoid non-linearity, is not sufficient to learn a sufficiently complex decision boundary in such a high dimensional space. As such, the model's loss and accuracy values converge but do not improve any further.

[1]BCEWithLogitsLoss(): https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

Single hidden layer shallow network

Next, we use a single hidden layer fully connected network (FCN) for the gender classification task. In order to choose the optimal number of neurons in the hidden layer, we train multiple models, each with a different possible number of neurons in the hidden layer. After that, based on each such trained model's performance on the validation dataset, we will choose the best performing model and use it to evaluate on the test dataset. Note that we are not peeking at the test set performance of each of these models and we select the best model based on validation accuracy alone.

We train 4 FCN models with one of these possible numbers of hidden layers: {50, 100, 500, 1000}. The validation performance of all these models is as shown in Table 4 and Figure 2.

Number of neurons in the hidden layer of the FCN	Validation accuracy
50	0.8434
100	0.8229
500	0.8377
1000	0.8411

Table 4: Validation accuracies of all the FCN models. The model with the highest validation accuracy has been highlighted.

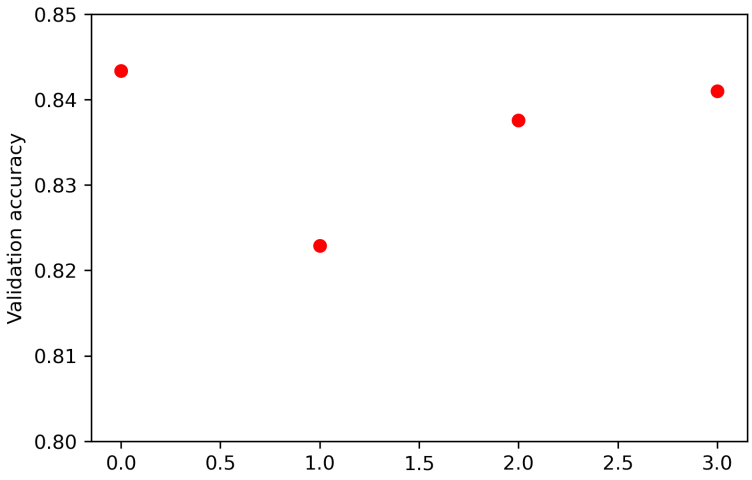

Figure 2: Visualizing the validation accuracies of all the FCN models with different numbers of neurons in the hidden layers.

We then report the best model's performance on the test dataset. As we can see in Table 4, the model with 50 neurons in the hidden layer yielded the best validation accuracy and is therefore chosen as the final model.

In order to visualize the training process, we plot the loss values and the classification accuracy of the best FCN model over the course of training for all the datasets in Figure 3.

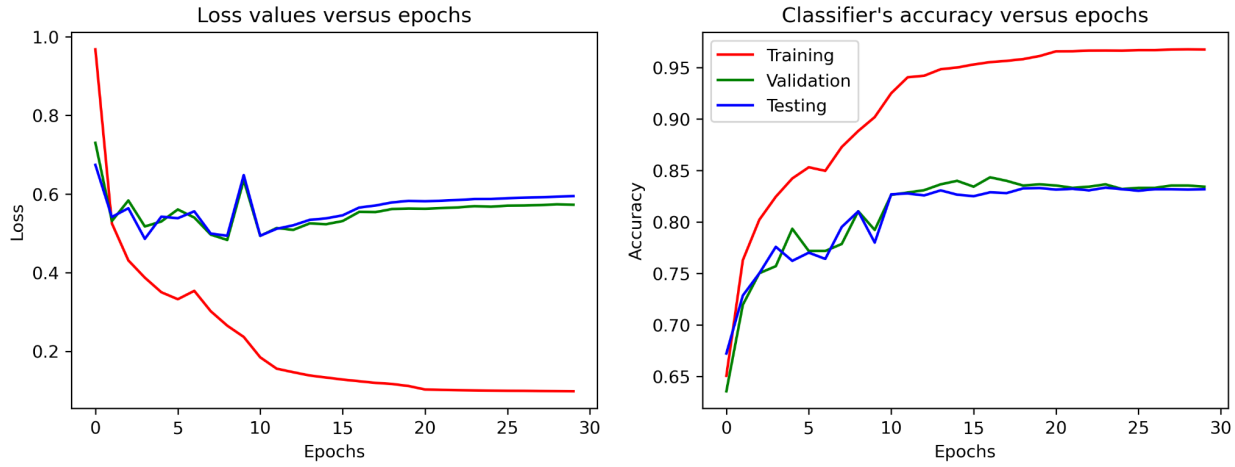

Figure 3: The best FCN model's loss and accuracy plots over the course of training.

As we can see in Figure 3, the loss values for all the dataset partitions converge. Similarly, the classifier's training accuracy approaches > 95% and is ≈82% for both the validation and the testing datasets. This indicates that the model has learned the decision boundary well, especially given the large number of parameters.

Figure 4 and Table 5 show the confusion matrix and the quantitative evaluation metrics of the best FCN model on the test set respectively.

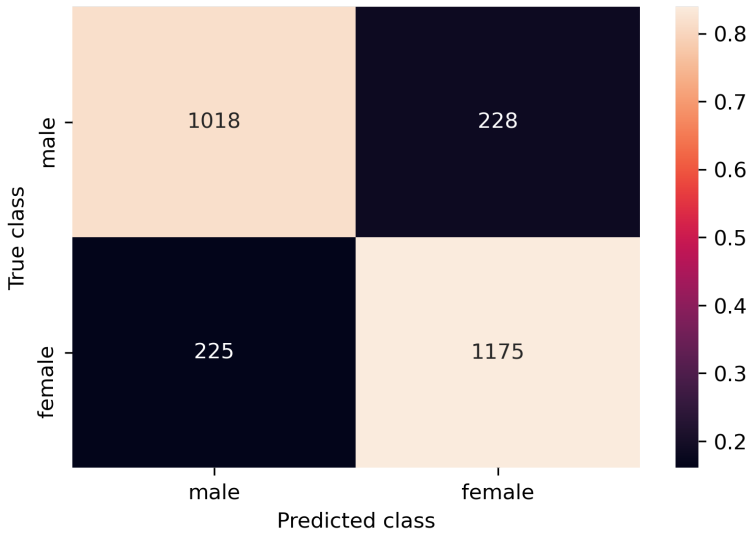

Figure 4: The confusion matrix of the best FCN model on the test set.

Gender	Precision	Recall	Overall Accuracy
male (0)	0.8190	0.8170	82.88%
female (1)	0.8375	0.8393	

Table 5: Quantitative metrics for the best gender classification FCN model.

Improving the generalization performance

In order to improve the generalization performance, we add data augmentation in the form of geometric transformations to the training images as well as regularization for the fully connected layers in the form of dropout.

For the data augmentation, we rotate the images by an angle randomly chosen in the range [-10°, 10°] followed by a random horizontal flip with a probability of 0.5.

Dropout is justified since the number of parameters of this model is orders of magnitude larger than the number of training images, which means that it is easy for the model to overfit to the training data. Using this dropout layer, we zero out values with a probability of 0.1.

In order to visualize the training process, we plot the loss values and the classification accuracy of this improved FCN model over the course of training for all the datasets in Figure 5.

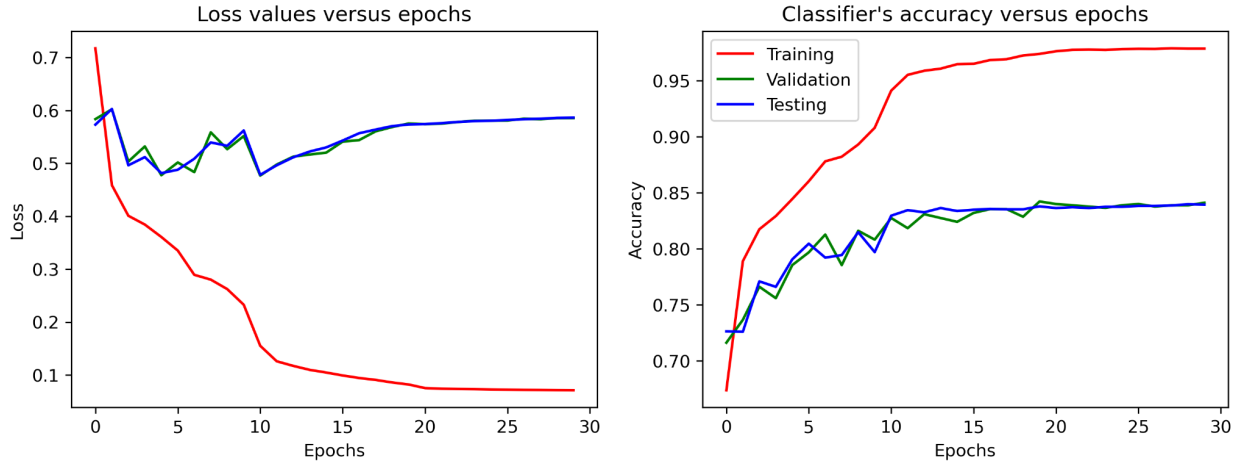

Figure 5: The improved FCN model's loss and accuracy plots over the course of training.

As we can see in Figure 5, the loss values for all the dataset partitions converge to lower values than before. Similarly, the classifier's training accuracy approaches > 95% and is ≈82% for both the validation and the testing datasets. This indicates that the model has learned well.

Figure 6 and Table 6 show the confusion matrix and the quantitative evaluation metrics of the improved FCN model on the test set respectively. We observe that the improved model misclassified fewer images (215 + 214 versus 228 + 225) than the original best FCN model, thus leading to an improved overall classification accuracy (83.79% versus 82.88%) and improved class-wise precision and recall.

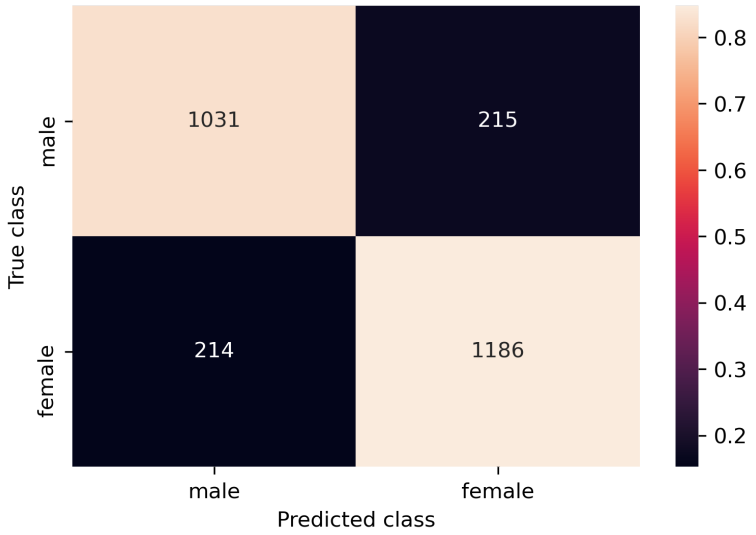

Figure 6: The confusion matrix of the improved FCN model on the test set.

Gender	Precision	Recall	Overall Accuracy
male (0)	0.8281	0.8275	83.79%
female (1)	0.8465	0.8471	

Table 6: Quantitative metrics for the improved gender classification FCN model.

Convolutional neural network

Finally, we use a convolutional neural network (FCN) for the gender classification task. In order to choose the optimal number of filters (or channels), the size of the filters, and the number of neurons in the fully connected layer, we train multiple models, each with a different possible combination of these hyperparameters. After that, based on each such trained model's performance on the validation dataset, we will choose the best performing model and use it to evaluate on the test dataset. Note that we are not peeking at the test set performance of each of these models and we select the best model based on validation accuracy alone.

An architectural choice that we make is that we set the number of channels in the second convolutional layer to be twice that of the number of channels in the first layer. This is a popular

CNN design choice and has been used by several widely used architectures such as U-Net, VGG-16, etc. Table 7 lists all the possible network architecture-related hyperparameter values that we use to train models with.

Hyperparameter	**Set of possible values**
kernel size of the 1st convolutional layer **(filter_size_1)**	{3, 5}
kernel size of the 2nd convolutional layer **(filter_size_2)**	{3, 5}
number of channels in the 1st convolutional layer **(n_channels_1)**	{4, 6, 8, 16}
number of neurons in the fully connected layer **(n_fc)**	{100, 500}

Table 7: Set of hyperparameter values to choose from.

Model	**Validation Accuracy**	**Model**	**Validation Accuracy**
CNN_3_3_4_100	0.8513	CNN_5_5_8_500	0.8683
CNN_3_3_4_500	0.8638	CNN_3_3_16_500	0.8695
CNN_3_5_4_100	0.8751	CNN_3_5_16_100	0.8854
CNN_3_5_4_500	0.8831	CNN_3_5_16_500	0.8763
CNN_5_3_4_100	0.8604	CNN_5_3_16_100	0.8717
CNN_5_3_4_500	0.8445	CNN_5_3_16_500	0.8774
CNN_5_5_4_100	0.8695	CNN_5_5_16_100	0.8910
CNN_5_5_4_500	0.8706	CNN_5_5_16_500	0.8956
CNN_3_3_8_100	0.8593	CNN_3_3_6_100	0.8649
CNN_3_3_16_100	0.8627	CNN_3_3_6_500	0.8638
CNN_3_3_8_500	0.8740	CNN_3_5_6_100	0.8763
CNN_3_5_8_100	0.8717	CNN_3_5_6_500	0.8842
CNN_3_5_8_500	0.8751	CNN_5_3_6_100	0.8717
CNN_5_3_8_100	0.8604	CNN_5_3_6_500	0.8695
CNN_5_3_8_500	0.8763	CNN_5_5_6_100	0.8797
CNN_5_5_8_100	0.8751	CNN_5_5_6_500	0.8717

Table 8: Validation accuracies of all the CNN models. The model with the highest validation accuracy has been highlighted.

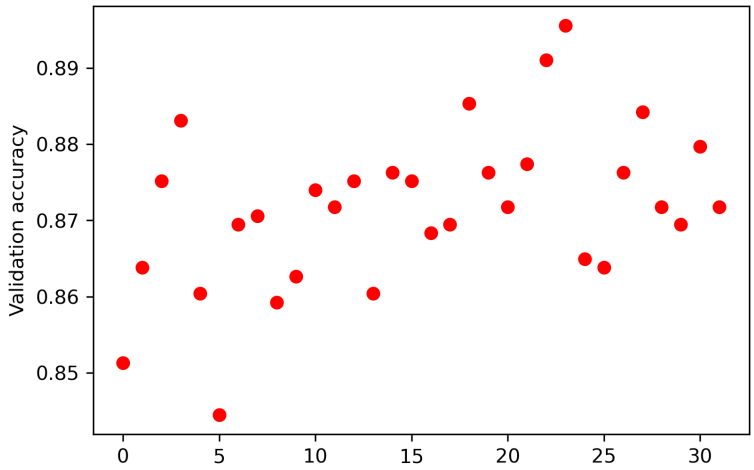

Figure 7: Visualizing the validation accuracies of all the CNN models with different numbers of neurons in the hidden layers.

As a result, we train 32 models, one with each hyperparameter combination from the set of values above. Table 8 and Figure 7 show the validation accuracies of all these models. The models are named as **CNN_<filter_size_1>_<filter_size_2>_<n_channels_1>_<n_fc>**.

We then report the best model's performance on the test dataset. As we can see in Table 8, the model with kernel sizes of 5 x 5 in both the convolutional layers, 16 channels in the first (and the second) convolutional layer, and 500 neurons in the fully connected layer yielded the best validation accuracy and is therefore chosen as the final model.

In order to visualize the training process, we plot the loss values and the classification accuracy of the best FCN model over the course of training for all the datasets in Figure 8.

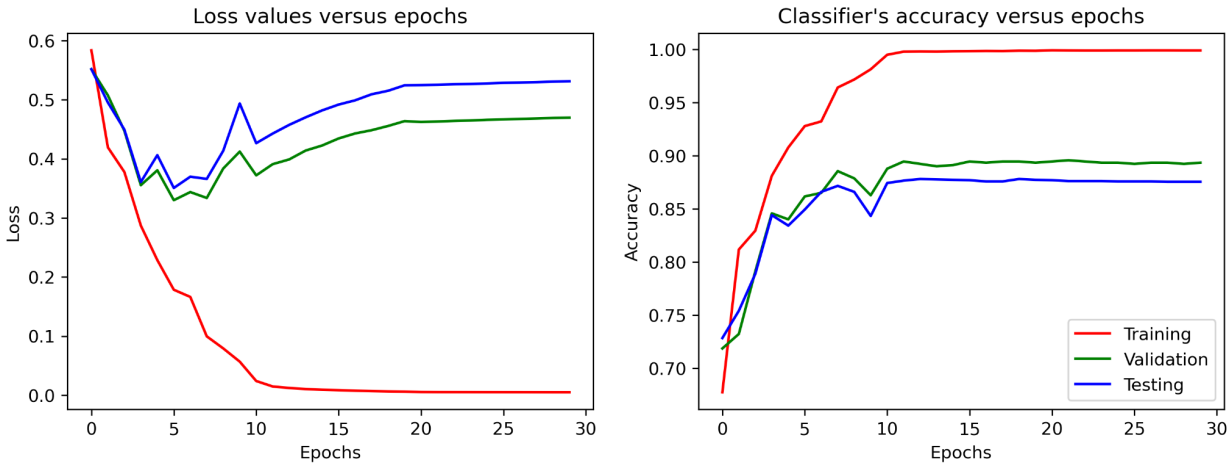

Figure 8: The best FCN model's loss and accuracy plots over the course of training.

As we can see in Figure 8, the loss values for all the dataset partitions converge well. Similarly, the classifier's training accuracy approaches > 95% and is ≈87% for both the validation and the testing datasets. This indicates that the model has learned the decision boundary well.

Figure 9 and Table 9 show the confusion matrix and the quantitative evaluation metrics of the best FCN model on the test set respectively.

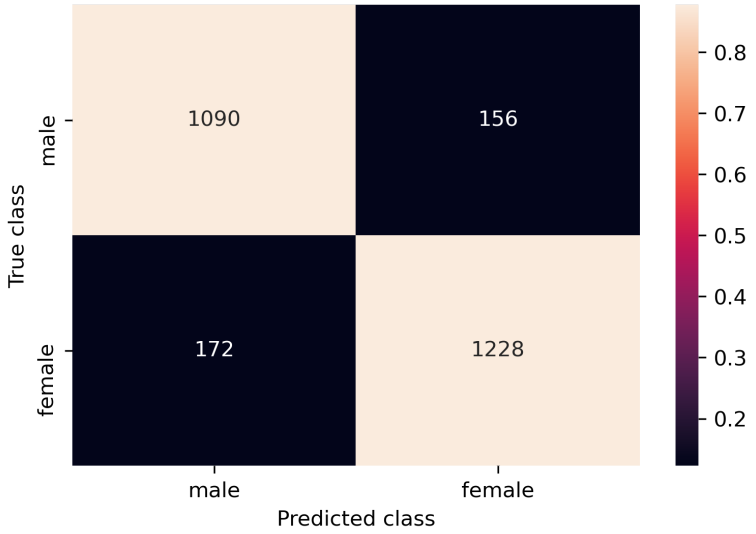

Figure 9: The confusion matrix of the best CNN model on the test set.

Gender	Precision	Recall	Overall Accuracy
male (0)	0.8637	0.8748	87.60%
female (1)	0.8873	0.8771	

Table 9: Quantitative metrics for the best gender classification CNN model.

Improving the generalization performance

In order to improve the generalization performance, we add data augmentation in the form of geometric transformations to the training images as we did with FCN models above. Given that we are using convolutional neural networks which are able to learn increasingly complex spatial patterns in images as we use more layers, increasing the variety of inputs would lead to the model learning more diverse sets of features.

For the data augmentation, we rotate the images by an angle randomly chosen in the range [-10°, 10°] followed by a random horizontal flip with a probability of 0.5.

In order to visualize the training process, we plot the loss values and the classification accuracy of this improved CNN model over the course of training for all the datasets in Figure 10.

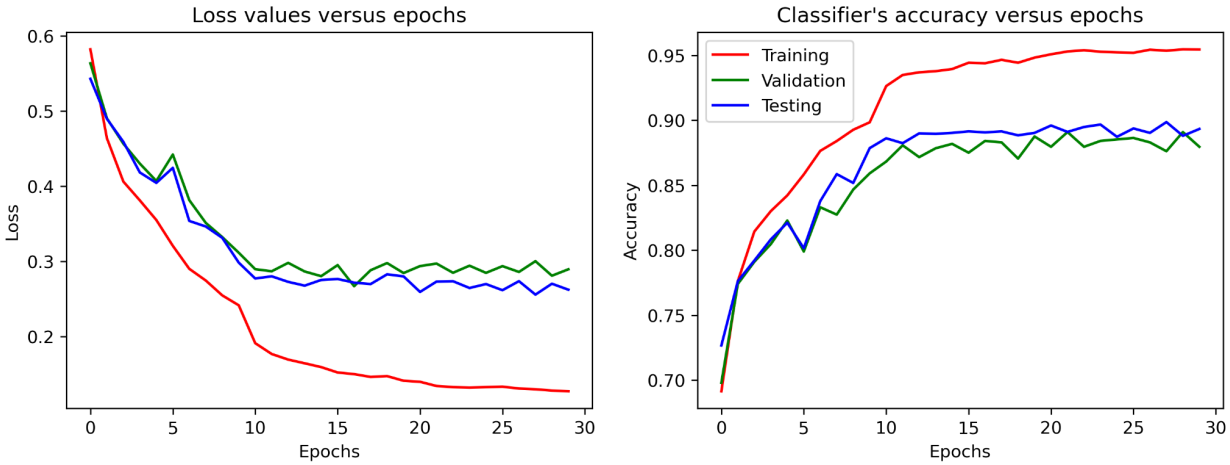

Figure 10: The improved CNN model's loss and accuracy plots over the course of training.

As we can see in Figure 10, the loss values for all the dataset partitions converge much better than any model before. Similarly, the classifier's training accuracy approaches > 95% and is ≈88% for both the validation and the testing datasets. This indicates that the model has learned quite well in only 30 training epochs.

Figure 11 and Table 10 show the confusion matrix and the quantitative evaluation metrics of the improved CNN model on the test set respectively. We observe that the improved model misclassified fewer images (124 + 150 versus 156 + 172) than the original best FCN model, thus leading to an considerably improved overall classification accuracy (89.65% versus 87.60%) and improved class-wise precision and recall.

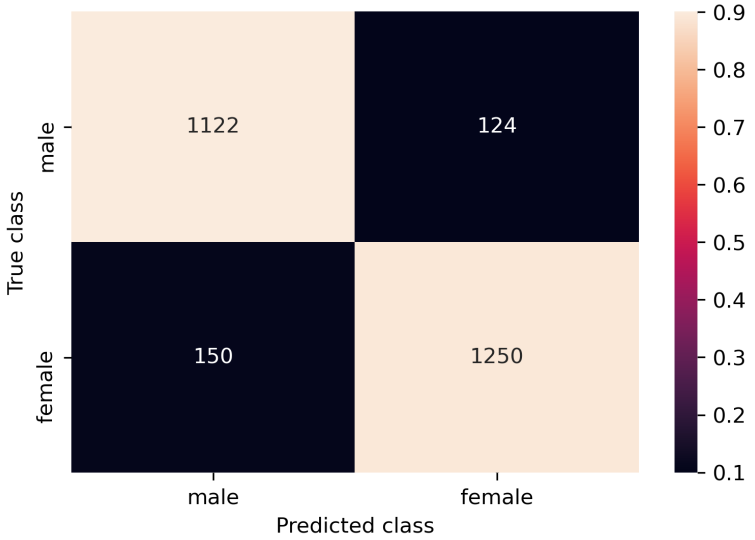

Figure 11: The confusion matrix of the improved CNN model on the test set.

Gender	Precision	Recall	Overall Accuracy
male (0)	0.8821	0.9005	89.65%
female (1)	0.9098	0.8929	

Table 10: Quantitative metrics for the improved gender classification CNN model.