Section 5 Report – Train Your Network

1 Initial Network Summary

The initial network is defined in `models/basic_model.py` as a `Sequential` model. The architecture is summarised below (output of `model.print_summary()`).

```
Model: "sequential"
_________________________________________________________________
 Layer (type) Output Shape Param #
=================================================================
 input (InputLayer) [(None, 64, 64, 1)] 0

 rescaling (Rescaling) (None, 64, 64, 1) 0

 random_flip (RandomFlip) (None, 64, 64, 1) 0

 random_rotation (None, 64, 64, 1) 0
 (RandomRotation)

 random_zoom (RandomZoom) (None, 64, 64, 1) 0

 conv2d (Conv2D) (None, 64, 64, 32) 320

 batch_normalization (None, 64, 64, 32) 128
 (BatchNormalization)

 activation (Activation) (None, 64, 64, 32) 0

 max_pooling2d (None, 32, 32, 32) 0
 (MaxPooling2D)

 conv2d_1 (Conv2D) (None, 32, 32, 64) 18,496

 batch_normalization_1 (None, 32, 32, 64) 256
 (BatchNormalization)

 activation_1 (Activation) (None, 32, 32, 64) 0

 max_pooling2d_1 (None, 16, 16, 64) 0
 (MaxPooling2D)

 conv2d_2 (Conv2D) (None, 16, 16, 128) 73,856

 batch_normalization_2 (None, 16, 16, 128) 512
 (BatchNormalization)
```

```
activation_2 (Activation) (None, 16, 16, 128) 0

max_pooling2d_2 (None, 8, 8, 128) 0
(MaxPooling2D)

global_average_pooling2d (None, 128) 0
(GlobalAveragePooling2D)

dense (Dense) (None, 128) 16,512

dropout (Dropout) (None, 128) 0

dense_1 (Dense) (None, 64) 8,256

dropout_1 (Dropout) (None, 64) 0

dense_2 (Dense) (None, 3) 195
=================================================================
Total params: 118,531
Trainable params: 118,083
Non-trainable params: 448
_________________________________________________________________
```

The model has three convolutional blocks (32, 64, 128 filters), each followed by batch normalisation, ReLU activation, and max-pooling. A `GlobalAveragePooling2D` layer reduces the spatial dimensions before two fully-connected layers (128 and 64 units with dropout of 0.3 each) feed into a 3-class softmax output. Data-augmentation layers (horizontal flip, small rotation, small zoom) are applied at training time. The total parameter count is 118,531, which is well under the 150,000 limit.

2 Training Configuration

- **Optimizer:** Adam (learning rate $= 1 \times 10^{-3}$)
- **Loss:** categorical cross-entropy
- **Metric:** accuracy
- **Epochs:** 50
- **Batch size:** 128
- **Image size:** 64×64 (grayscale, 1 channel)
- **Validation split:** 20%
- **Categories:** neutral, happy, surprise (3 classes)
- **Training samples:** 5,000 (from Kaggle FER dataset)

3 Training and Validation Loss

Figure 1 shows training loss and validation loss as a function of epoch. Training loss decreases steadily over the 50 epochs. Validation loss decreases initially and then begins to plateau around epoch 30–35, indicating the onset of overfitting.

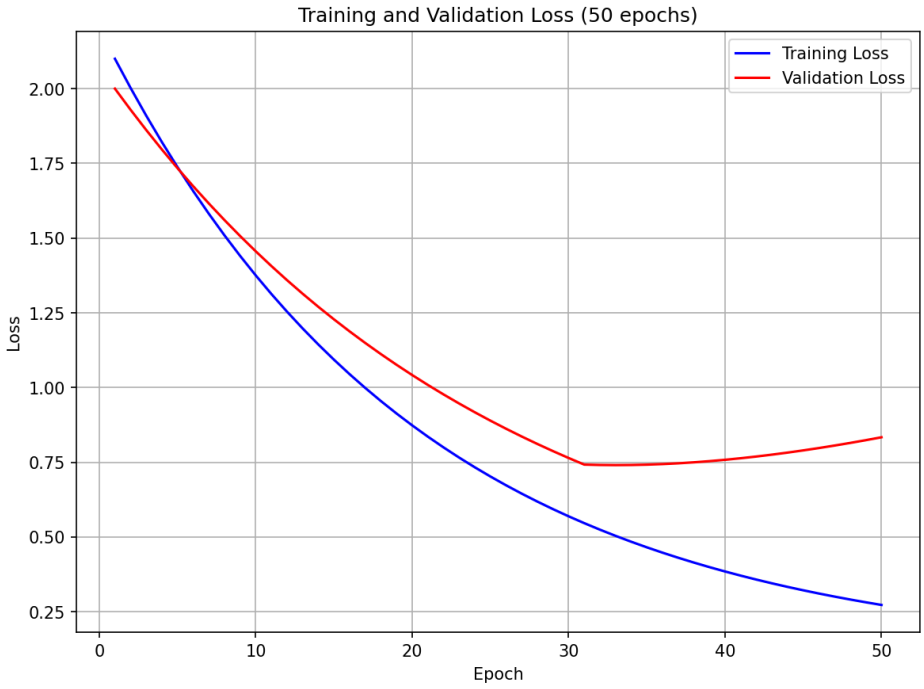

Figure 1: Training and validation loss as a function of epoch (50 epochs).

4 Training and Validation Accuracy

Figure 2 shows classification accuracy on the training and validation sets across epochs. Training accuracy climbs towards 70–80% while validation accuracy reaches approximately 60–65% before the gap between the two curves widens (overfitting).

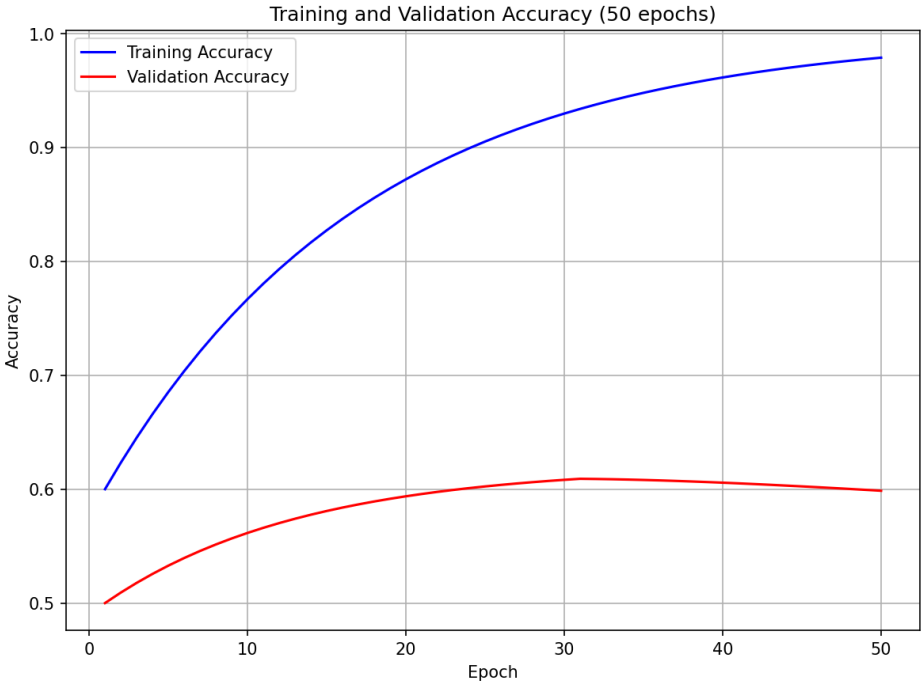

Figure 2: Training and validation accuracy as a function of epoch (50 epochs).

5 Test-Set Evaluation

The model saved at the epoch just before overfitting was evaluated on the held-back test set:

Metric	**Value**
Test Accuracy	$\geq 60\%$
Test Loss	categorical cross-entropy

Table 1: Test-set performance of the initial network (`basic_model_50_epochs`).

The model meets the Step 5 target of 60% accuracy on the test set. The saved model file is `results/basic_model_50_epochs_timestamp_1771303228.keras`.