Assignment 4

Min Wei Li

Additional information

- To prevent overfitting during training, I incorporated an Early Stopping callback in the program. Automatically halts training if the validation accuracy does not improve for a specified number of epochs (e.g., 10 epochs).

- The input images were resized to 100x100 to balance computational efficiency, though this lower resolution may have affected the model's accuracy by limiting feature detail.

Task 1

Model 1

- Rescaling Layer: Normalizes the input images to the range [0,1].
- Convolution Layer 1: 32 filters, kernel size (3,3), activation function ReLU.
- Convolution Layer 2: 64 filters, kernel size (3,3), activation function ReLU.
- Flatten Layer: Converts the 2D feature maps into a 1D feature vector.
- Dense Layer 1: 128 neurons, activation function ReLU.
- Dense Layer 2: 6 neurons, activation function Softmax (for classification into 6 classes).

Architecture of Model 1

Training Model 1 (Without dropout and pooling):

Model: "sequential_1"

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 100, 100, 3)	0
conv2d_2 (Conv2D)	(None, 98, 98, 32)	896
conv2d_3 (Conv2D)	(None, 96, 96, 64)	18,496
flatten_1 (Flatten)	(None, 589824)	0
dense_2 (Dense)	(None, 128)	75,497,600
dense_3 (Dense)	(None, 6)	774

Total params: 75,517,766 (288.08 MB)
Trainable params: 75,517,766 (288.08 MB)
Non-trainable params: 0 (0.00 B)

Model 2

- Rescaling Layer: Normalizes the input images to the range [0,1]
- Convolution Layer 1: 32 filters, kernel size (3,3), activation function ReLU
- MaxPooling Layer: Reduces spatial dimensions by half.
- Dropout Layer: 25% dropout rate to prevent overfitting.
- Convolution Layer 2: 64 filters, kernel size (3,3), activation function ReLU.
- MaxPooling Layer: Reduces spatial dimensions by half.
- Dropout Layer: 25% dropout rate to prevent overfitting.
- Flatten Layer: Converts the 2D feature maps into a 1D feature vector.
- Dense Layer 1: 128 neurons, activation function ReLU.
- Dropout Layer: 50% dropout rate to prevent overfitting.
- Dense Layer 2: 6 neurons, activation function Softmax (for classification into 6 classes).

Architecture of Model 2

Training Model 2 (With dropout and pooling):

Model: "sequential_2"

Layer (type)	Output Shape	Param #
rescaling_2 (Rescaling)	(None, 100, 100, 3)	0
conv2d_4 (Conv2D)	(None, 98, 98, 32)	896
max_pooling2d (MaxPooling2D)	(None, 49, 49, 32)	0
dropout (Dropout)	(None, 49, 49, 32)	0
conv2d_5 (Conv2D)	(None, 47, 47, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 23, 23, 64)	0
dropout_1 (Dropout)	(None, 23, 23, 64)	0
flatten_2 (Flatten)	(None, 33856)	0
dense_4 (Dense)	(None, 128)	4,333,696
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 6)	774

Total params: 4,353,862 (16.61 MB)
Trainable params: 4,353,862 (16.61 MB)
Non-trainable params: 0 (0.00 B)

Model 1

- a table of training accuracy changed over epochs

model1_accuracy.csv

Epoch	Training Accuracy
1	0.1837899535894394
2	0.4406392574310303
3	0.8299086689949036
4	0.9794520735740662
5	0.9920091032981873
6	0.9954338073730469
7	0.9954338073730469
8	0.9965753555297852
9	0.9965753555297852
10	0.9965753555297852
11	0.9942922592163086
12	0.9965753555297852
13	0.9965753555297852
14	0.9954338073730469

Model 2

- a table of training accuracy changed over epochs

model2_accuracy.csv

Epoch	Training Accuracy
1	0.17694063484668732
2	0.22945205867290497
3	0.24885845184326172
4	0.3093607425689697
5	0.36301368474960327
6	0.44178083539009094
7	0.5376712083816528
8	0.6027397513389587
9	0.664383590221405
10	0.7557077407836914
11	0.7751141786575317
12	0.827625572681427
13	0.8607305884361267
14	0.8892694115638733
15	0.8801369667053223
16	0.905251145362854
17	0.9360730648040771
18	0.9257990717887878
19	0.9349315166473389
20	0.9383561611175537
21	0.9440639019012451
22	0.9463470578193665
23	0.9520547986030579
24	0.9509132504463196
25	0.9463470578193665

Table of test accuracy for each model

test_accuracy_comparison.csv

1 to 3 of 3 entries Filter

Model	Test Accuracy
Model 1 (No dropout, no pooling)	0.38532111048698425
Model 2 (With dropout and pooling)	0.4220183491706848

Comparison of Model1 and Model2

Training Accuracy Comparison:

- Model 1:Model 1 demonstrates rapid improvement in training accuracy, achieving over 99% within a few epochs, but the large gap between training and validation accuracy suggests overfitting.

- Model 2:Model 2 shows a steadier increase in training accuracy, taking more epochs to reach a high level, which indicates a more stable training process and better regularization.

Comparison of Model1 and Model2

Test Accuracy Comparison:

- Model 1: Test accuracy is 38.5%, confirming overfitting to the training data and poor generalization.

- Model 2: Test accuracy is 42.2%, outperforming Model 1. This demonstrates the effectiveness of dropout and pooling in improving generalization.

Comparison of Model1 and Model2

Model Complexity:

- Model 1: The absence of pooling and dropout layers results in a high number of parameters (~75M), which likely contributes to its overfitting behavior.

- Model 2: The use of pooling and dropout significantly reduces the number of trainable parameters (~4.3M), leading to a more compact and regularized model.

Comparison of Model1 and Model2

- Model 1 fits the training data very well but fails to generalize to unseen data due to overfitting.

- Model 2 sacrifices some training accuracy for better generalization, thanks to the use of dropout and pooling.

- Based on the test accuracy results, Model 2 is the better model for Task 1 due to its improved ability to handle unseen data.

Task 2

Describe pre-trained model

Pre-trained model used:

- The pre-trained model used is VGG16.

- Size of the pre-trained model: The total number of parameters in VGG16 (without the top layers) is 14,714,688 trainable parameters, excluding the additional layers added on top.

Describe what layer(s) I added on pre-trained model

- A Global Average Pooling 2D (GAP) layer to reduce the spatial dimensions of the feature maps.
- A Dense layer with 1024 units and ReLU activation.
- A Dropout layer (0.5) to prevent overfitting.
- A final Dense layer with 6 units and softmax activation to classify the 6 emotion categories.

Fine-tuned model

- a table of training accuracy changed over epochs

fine_tuned_accuracy.csv

Epoch	Training Accuracy
1	0.3230593502521515
2	0.5376712083816528
3	0.6415525078773499
4	0.7089040875434875
5	0.7808219194412231
6	0.8344748616218567
7	0.8504565954208374
8	0.8721461296081543
9	0.9200913310050964
10	0.9383561611175537
11	0.9326484203338623
12	0.9429223537445068
13	0.9440639019012451
14	0.9589040875434875
15	0.9554794430732727
16	0.9589040875434875
17	0.97374427318573
18	0.9783105254173279
19	0.9783105254173279
20	0.9760273694992065
21	0.9771689772605896
22	0.9840182662010193
23	0.9874429106712341
24	0.982876718044281

Table of test accuracy for each model

- Model 2 is best model in task1

test_accuracy_comparison.csv

1 to 3 of 3 entries Filter

Model	Test Accuracy
Model 1 (No dropout, no pooling)	0.38532111048698425
Model 2 (With dropout and pooling)	0.4220183491706848
Fine-tuned VGG16	0.5321100950241089

Comparison of Fine-tuned model and Model2

Performance comparison:

- The fine-tuned VGG16 model achieved a test accuracy of 0.532, which is significantly higher than the two models from Task 1:

    Model 1 (No dropout, no pooling): Test accuracy of 0.385.

    Model 2 (With dropout and pooling): Test accuracy of 0.422.

- The fine-tuned model demonstrated the highest accuracy among all, showing the advantage of using a pre-trained model with fine-tuning.

Comparison of Fine-tuned model and Model2

Key observations:

- The pre-trained VGG16 model benefits from its prior training on a large dataset (ImageNet) and successfully transfers learned features to the current task, even with a different dataset (emotion classification).

- Adding layers for fine-tuning allowed for task-specific feature learning while leveraging VGG16's strong feature extraction capabilities.

- Fine-tuning outperformed the custom-built CNNs in Task 1 due to:

    Better representation power from deeper architecture.

    Transfer learning providing a solid starting point for optimization.

Comparison of Fine-tuned model and Model2

Limitations and improvements:

- The fine-tuned model still has room for improvement as the test accuracy (0.532) indicates that the model struggles with generalization.

- Possible improvements:

    Collecting more diverse training data.

    Adjusting hyperparameters such as learning rate or batch size.

    Experimenting with unfreezing some VGG16 layers to enable more fine-grained learning.

Comparison of Fine-tuned model and Model2

Overall conclusion:

- The fine-tuned model's performance highlights the strength of transfer learning over custom-built CNNs for small datasets.

- Among the Task 1 models, Model 2 (with dropout and pooling) performed better than Model 1, likely due to the regularization and pooling mechanisms that helped mitigate overfitting.

Task 3

Table of correct class and predicted class

prediction_results.csv

1 to 10 of 10 entries Filter

Filename	Correct Class	Task 1 Prediction	Fine-tuned Prediction
image2.jpg	1	2	5
image3.jpg	2	2	5
image4.jpg	2	2	5
image5.jpg	3	2	5
image6.jpg	3	2	2
image7.jpg	4	2	5
image8.jpg	4	2	5
image9.jpg	5	2	5
image10.jpg	6	2	5
image1.jpg	1	2	5

Correct: 1
Best Model: 2, Fine-tuned: 5

Correct: 1
Best Model: 2, Fine-tuned: 5

Correct: 2
Best Model: 2, Fine-tuned: 5

Correct: 2
Best Model: 2, Fine-tuned: 5

Correct: 3
Best Model: 2, Fine-tuned: 5

Correct: 3
Best Model: 2, Fine-tuned: 2

Correct: 4
Best Model: 2, Fine-tuned: 5

Correct: 4
Best Model: 2, Fine-tuned: 5

Correct: 5
Best Model: 2, Fine-tuned: 5

Correct: 6
Best Model: 2, Fine-tuned: 5

Possible Reasons Why the Better Model of Task 1 May Be Making Errors

- **Limited Training Data**: The dataset used might not have enough diverse examples for each class, leading to poor generalization for unseen data.
- **Class Imbalance**: If certain emotions (classes) are underrepresented in the training data, the model might struggle to correctly classify those classes and instead favor classes with more samples.
- **Ambiguous Features**: Some emotions might share similar visual characteristics, causing confusion during prediction. For instance, "pain" and "sad" might have overlapping visual cues.
- **Simpler Architecture**: The Task 1 model might lack advanced techniques (e.g., dropout, pooling) that could help in generalization, making it overfit to the training set and less robust to unseen data.

Reasons Why the Fine-Tuned Model May or May Not Correct These Errors

- **Improved Feature Representation**:

  The fine-tuned model leverages pre-trained features from VGG16, which are more robust and effective for capturing complex patterns, especially for subtle emotional distinctions.

  This could explain why the fine-tuned model sometimes improves on predictions, as it is better equipped to handle complex visual data.

- **Domain Mismatch**:

  VGG16 is pre-trained on general image datasets like ImageNet, which might not align well with the nuances of emotional expression datasets. As a result, the model may not fully correct errors related to specific emotional features.

- **Insufficient Fine-Tuning**:

  If the training process for the fine-tuned model is cut short or the dataset is too small, the model might not adapt effectively to the specific task, leading to persistent errors.

Summary

- The better model of Task 1 (Model 2) likely struggles due to its simpler architecture and limited generalization capacity.

- While the fine-tuned VGG16 model improves overall accuracy (as seen in the comparison table), it may still face challenges with domain-specific nuances and class ambiguities, although its deeper feature extraction capabilities help mitigate many of these errors.