1.1 Task A: Model Selection

Because in many cases, people decide to use transfer learning, they face low data quantity and niche vertical areas data that cause difficulty in collecting data for training and no similar domain public datasets. Therefore, I removed 90% of the training and test data from the following dataset overview. And choose the pre-train models trained by data that differ from our data's domain. I choose two pre-train models, VGG-16 and MobileNetV2. VGG-16 has a simple architecture that helps us understand transfer learning processes. However, training VGG-16 is too slow if we use free Colab to train the model. That is why I chose MobileNetV2 as another pre-train model. We also can compare the larger model with a smaller model's performance on small dataset transfer learning problems.

Train Dataset Statistics: Number of Images: 76, Label Counts: {0: 38, 1: 38}
Test Dataset Statistics: Number of Images: 76, Label Counts: {0: 38, 1: 38}
Validation Dataset Statistics: Number of Images: 16, Label Counts: {0: 8, 1: 8}

1.2 Task B: Fine-tuning the ConvNet

In phase 1, I only modified the last layer of the classifier in both VGG-16 and MobileNetV2 models using the following structure. And only tuning the classifier's parameters without tuning the Cov layer's parameters.

VGG-16 **(Code: https://github.com/TingsGithub/NTHU_2024_DLBOI_HW/blob/main/HW4/hw4_VGG_16_Done.ipynb)**

```
(classifier): Sequential(
   (0): Linear(in_features=25088, out_features=4096, bias=True)
   (1): ReLU(inplace=True)
   (2): Dropout(p=0.5, inplace=False)
   (3): Linear(in_features=4096, out_features=4096, bias=True)
   (4): ReLU(inplace=True)
   (5): Dropout(p=0.5, inplace=False)
   (6): Sequential(
     (0): Linear(in_features=4096, out_features=1, bias=True)
     (1): Sigmoid())
```

MobileNetV2 **(Code: https://github.com/TingsGithub/NTHU_2024_DLBOI_HW/blob/main/HW4/hw4_MobileNet_V2_Done.ipynb)**

```
(classifier): Sequential(
   (0): Dropout(p=0.2, inplace=False)
   (1): Sequential(
     (0): Linear(in_features=1280, out_features=1, bias=True)
     (1): Sigmoid())
```

1.3 Task C: ConvNet as Fixed Feature Extractor

In the following table, I apply the Design of Experiments (DOE) Full Factorial Design (FFD), the same as HW3. Every RunOrder training 10 epochs.

StdOrder	RunOrder	CenterPt	Blocks	lr	weight_decay	factor	criterion
8	1	1	1	0.001	0.001	0.1	nn.MSELoss()
2	2	1	1	0.001	0.0001	0.01	nn.MSELoss()
16	3	1	1	0.001	0.001	0.1	nn.BCELoss()
6	4	1	1	0.001	0.0001	0.1	nn.MSELoss()
13	5	1	1	0.0001	0.0001	0.1	nn.BCELoss()
1	6	1	1	0.0001	0.0001	0.01	nn.MSELoss()
3	7	1	1	0.0001	0.001	0.01	nn.MSELoss()
4	8	1	1	0.001	0.001	0.01	nn.MSELoss()
10	9	1	1	0.001	0.0001	0.01	nn.BCELoss()
7	10	1	1	0.0001	0.001	0.1	nn.MSELoss()
14	11	1	1	0.001	0.0001	0.1	nn.BCELoss()
11	12	1	1	0.0001	0.001	0.01	nn.BCELoss()
12	13	1	1	0.001	0.001	0.01	nn.BCELoss()
9	14	1	1	0.0001	0.0001	0.01	nn.BCELoss()
15	15	1	1	0.0001	0.001	0.1	nn.BCELoss()
5	16	1	1	0.0001	0.0001	0.1	nn.MSELoss()

1.4 Task D: Comparison and Analysis

However, even the best model after completing DOE FFD is the one that overfits the training data. And MobileNetV2 training result is bad. The following table shows the best result after completing the DOE FFD.

Model	RunOrder	Epoch	Train Loss	Train Accuracy	Val Loss	Val Accuracy	Test Accuracy
VGG-16	DOE RunOrder 16	10	0.03	97.37	0.08	87.50	0.65
MobileNetV2	DOE RunOrder 1	10	0.26	48.68	0.23	75.00	0.62

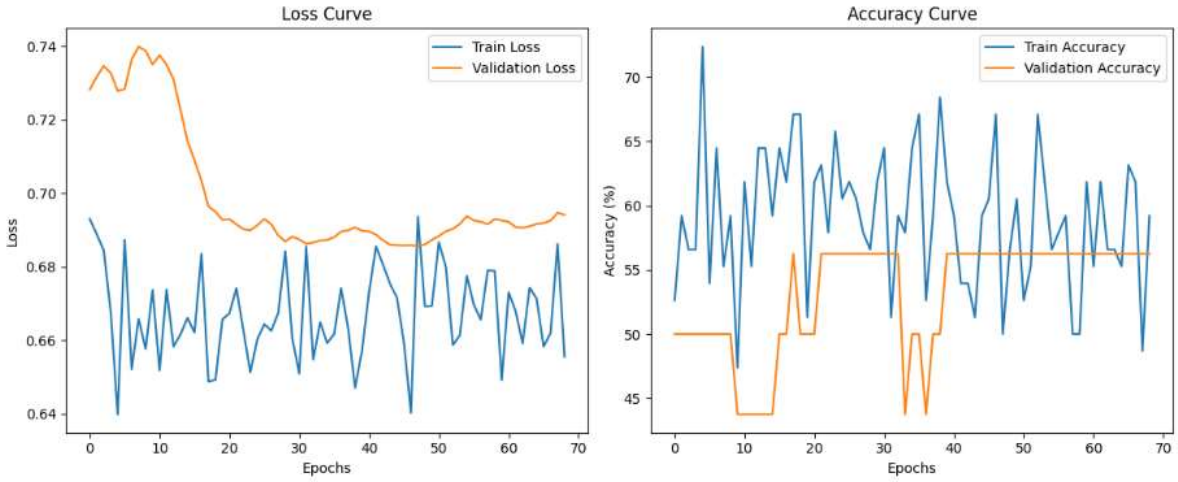
1.5 Task E: Test Dataset Analysis

Because the MobileNetV2 DOE result is bad. I try to optimize by finetuning hyperparameters but not modify the structure and train more epochs. The result is still bad.

Training only finetune hyperparameters and train more epochs:

(Code: https://github.com/TingsGithub/NTHU_2024_DLBOI_HW/blob/main/HW4/hw4_MobileNet_V2_Phase2_01_Done.ipynb)

Epoch 69/69 - loss: 0.6556 - train_acc: 59.21% - val_loss: 0.6942 - val_acc: 56.25%, Test Accuracy: 45.00%



I guess that caused by tunable parameters is insufficient because the VGG-16 model has more tunable parameters. I added a full connection layer to the classifier.

New MobileNetV2

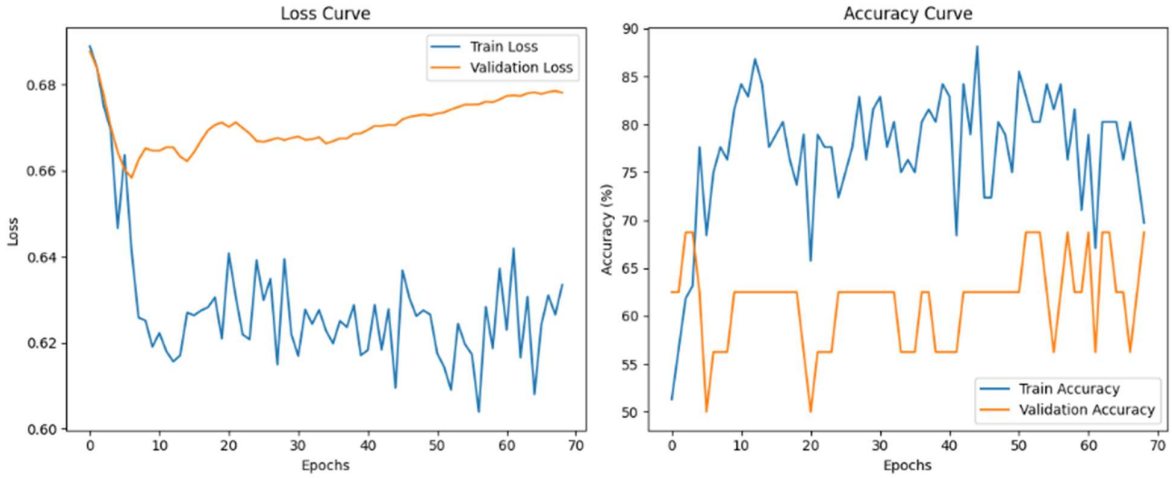
(classifier): Sequential(
  (0): Dropout(p=0.2, inplace=False)
  (1): Sequential(
    (0): Linear(in_features=1280, out_features=128, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=128, out_features=1, bias=True)
    (4): Sigmoid())

Then, retrain the new MobileNetV2 and have better results.

Add one full connection layer:

(Code: https://github.com/TingsGithub/NTHU_2024_DLBOI_HW/blob/main/HW4/hw4_MobileNet_V2_Phase2_02_Done.ipynb)

Epoch 69/69 - loss: 0.6334 - train_acc: 69.74% - val_loss: 0.6781 - val_acc: 68.75%, Test Accuracy: 62.50%

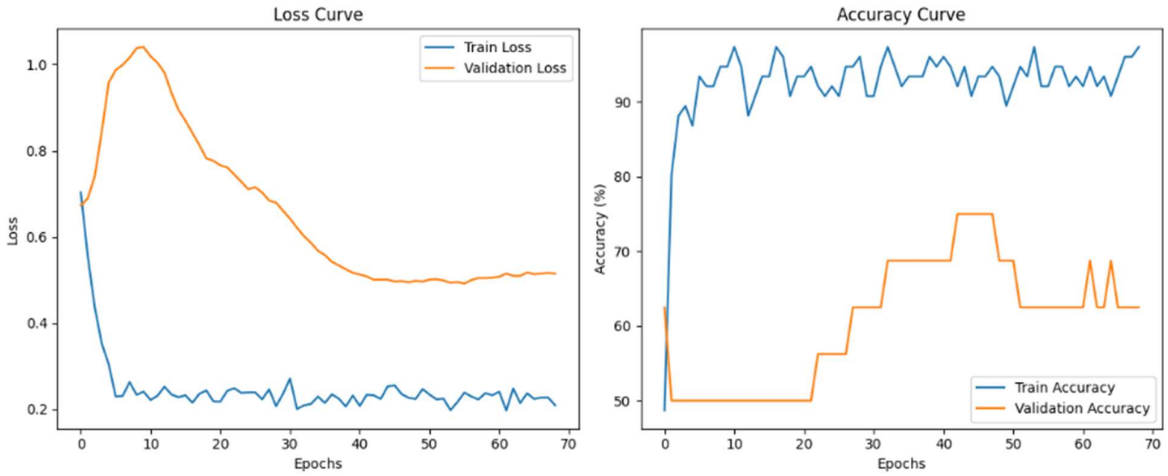


But we can improve more by retraining the entire model, including the Cov layer's filter. That means we can't only retrain the full connection layers. We need to retrain Cov filters so the model can learn how to detect features in new domain data.

Train entire model:

(Code: https://github.com/TingsGithub/NTHU_2024_DLBOI_HW/blob/main/HW4/hw4_MobileNet_V2_Phase2_01_Full_Train_Done.ipynb)

Epoch 69/69 - loss: 0.2089 - train_acc: 97.37% - val_loss: 0.5145 - val_acc: 62.50%, Test Accuracy: 75.00%



Conclusion

In this homework assignment. I simulated a real-world problem with difficulty having more data and could not find an open dataset like our data's domain. We need to try the transfer learning method and try to optimize performance. The results show that we need to retrain sufficient parameters, and we cannot only retrain fully connection layers. Because only retraining full connection layers makes the model easy to overfit to training data, especially small datasets. We must retrain the Cov layer's filters to teach the model to detect features in our data's domain.