Convolutional Neural Networks (CNNs)

In this section, we investigate the performance of Convolutional Neural Networks (CNNs), a fundamental model choice for image analysis tasks because of its ability to efficiently identify and interpret patterns and structures at multiple levels of detail within visual data.

CNNs are a type of deep learning model specifically designed for processing structured grid data, such as images, by leveraging convolutional layers to detect local patterns like edges, textures, and shapes. These patterns are then progressively combined through deeper layers to recognize complex features and objects within an image.

CNNs operate in two main phases: **feature extraction** and **classification**. In the feature extraction phase, convolutional layers and pooling layers work together to extract hierarchical features from the input image. Convolutional layers apply learnable filters to identify local patterns, while pooling layers reduce the spatial dimensions, retaining the most significant information while reducing computational complexity. This results in feature maps that represent the essential characteristics of the image.

In the classification phase, the extracted features are passed to fully connected layers, which act as a traditional neural network classifier. These layers interpret the extracted features and map them to output classes through activation functions like softmax. This combination of feature extraction and classification enables CNNs to excel at tasks such as image recognition and object detection.

To optimize the performance of our CNN model, we evaluated several parameters that influence accuracy, generalization, and computational efficiency. These parameters include dataset size, image resolution, learning rate, data augmentation techniques, and the depth of the CNN architecture (number of convolutional layers). By varying these parameters, we aimed to identify the configurations that maximize model performance for our image classification task.

Furthermore, we compared the custom-trained CNN with a transfer learning approach using a pre-trained VGG16 model. Transfer learning, which involves fine-tuning a pre-trained network on a new dataset, is often used to leverage the rich feature representations learned on large datasets like ImageNet. The detailed investigation on transfer learning approach is presented in the following section B.

Two-Layer CNN Model

This model consists of **two convolutional layers**, each followed by batch normalization and max pooling. The convolutional layers learn feature maps by applying 32 and 64 filters, respectively. The batch normalization layers help to stabilize and accelerate training by normalizing the feature maps. Max pooling is applied to reduce spatial dimensions, thereby decreasing computational complexity while retaining key features.

The flattened feature maps are then passed through a dense layer with 128 neurons and ReLU activation, followed by a dropout layer with a rate of 0.6 to prevent overfitting. Finally, the output layer, with 38 neurons and a softmax activation function, maps the features to the corresponding 38 classes (likely representing categories in your dataset).

Model Architecture:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
batch_normalization (BatchNormalization)	(None, 222, 222, 32)	128
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 109, 109, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
flatten (Flatten)	(None, 186624)	0
dense (Dense)	(None, 128)	23,888,000
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 38)	4,902

Total params: 23,912,678 (91.22 MB)
Trainable params: 23,912,486 (91.22 MB)
Non-trainable params: 192 (768.00 B)

Three-Layer CNN Model

The three-layer model expands on the two-layer architecture by adding an additional convolutional layer with 128 filters, batch normalization, and max pooling. This extra convolutional layer allows the model to extract deeper, more complex features, which can improve accuracy for datasets with high variability or fine-grained categories.

After the feature extraction layers, the flattened output is passed through a dense layer with 512 neurons, providing greater capacity to learn complex relationships. A dropout layer with a rate of 0.5 is used to reduce overfitting, and the final softmax layer maps the output to 38 classes.

Model Architecture:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
batch_normalization (BatchNormalization)	(None, 222, 222, 32)	128
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 109, 109, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 52, 52, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
flatten (Flatten)	(None, 86528)	0
dense (Dense)	(None, 512)	44,302,848
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 38)	19,494

Total params: 44,416,486 (169.44 MB)
Trainable params: 44,416,038 (169.43 MB)
Non-trainable params: 448 (1.75 KB)

Our findings highlight that larger datasets significantly enhance model accuracy, while smaller image sizes (224x224) work well for simpler architectures or higher learning rates, and larger image sizes (256x256) excel with deeper models and lower learning rates. Lower learning rates (0.00001) improve stability and generalization for deeper models, whereas higher learning rates (0.001) can cause instability, especially with larger resolutions. Data augmentation improves test accuracy and generalization but increases training time (~10.5 hours), and adding more CNN layers enhances validation and test accuracy with a manageable training time (~2 hours). The best-performing configuration, a 3-layer CNN with 256x256 image size and a learning rate of 0.00001, achieved the highest test accuracy (0.91) and validation loss (0.24), demonstrating strong generalization. The VGG16 model showed competitive accuracy (0.88). Below is the summary of our findings.

Key Insights:

1. **Dataset size:** Larger dataset size significantly boosts validation accuracy, highlighting the importance of sufficient training data.
2. **Image Size:**
    - Smaller image sizes (224x224) are better for simpler architectures or higher learning rates.
    - Larger image sizes (256x256) are better with deeper models and lower learning rates.
3. **Learning rate:**
    - Learning rate tuning has a significant impact on model accuracy.
    - A lower learning rate (0.00001) provides better stability and generalization for deeper models.
    - Higher learning rates (0.001) can lead to instability, especially with larger resolutions.
4. **Data augmentation:** improves generalization (higher test accuracy) but comes at the cost of increased running time (~10.5 h).
5. **Model depth:**Increasing CNN layers improves both validation and test accuracy with a manageable running time (~2 h).
6. VGG16 shows good performance. Further optimization will be presented in section B.

Ranking (Based on Performance Metrics and Efficiency):

1. **Best Configuration**: **K (256x256, 3xConv2D, lr=0.00001)**:

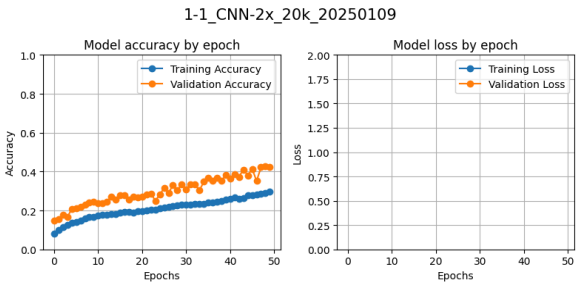
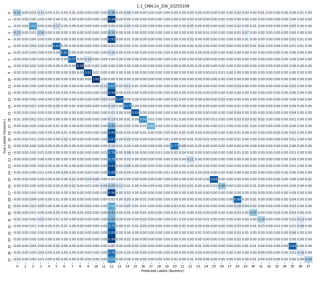
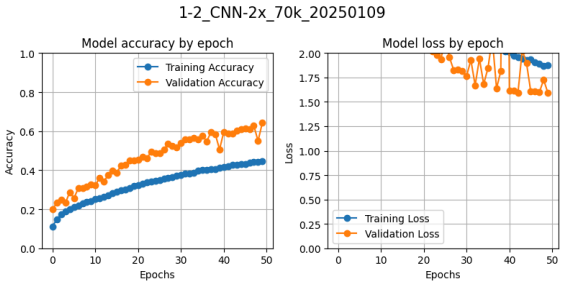
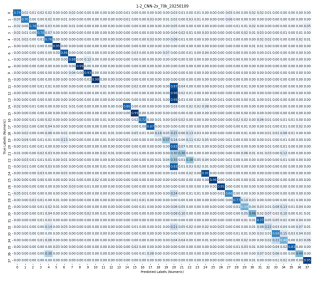
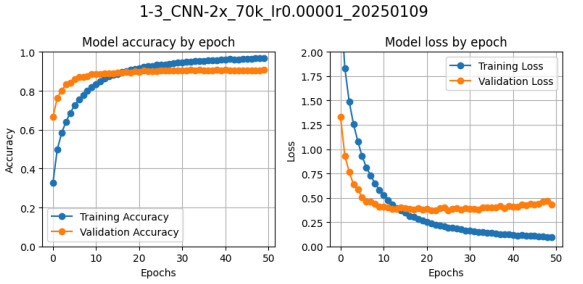
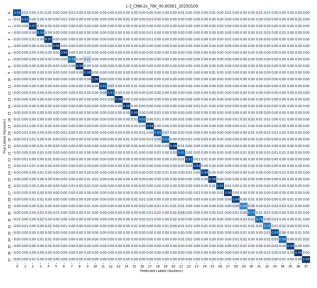
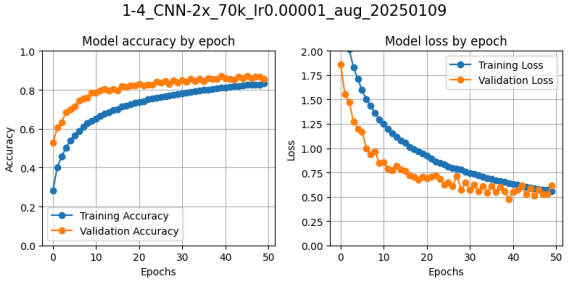
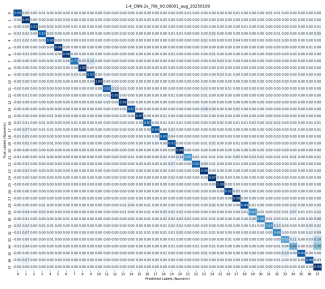
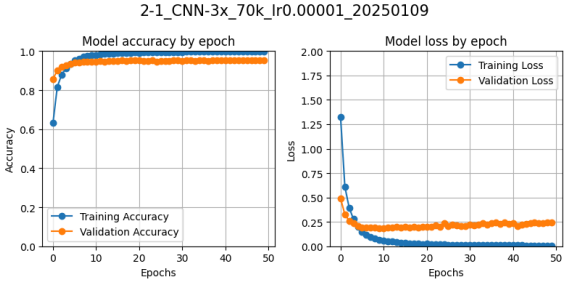
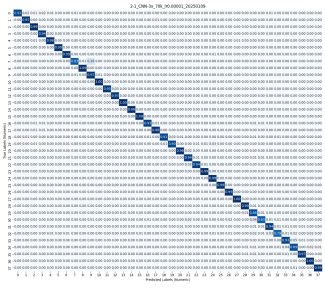
- **Test Accuracy**: 0.91
- **Validation Loss**: 0.24
- Demonstrates strong generalization with a deeper model and low learning rate.

2. **F (VGG16 Frozen)**:

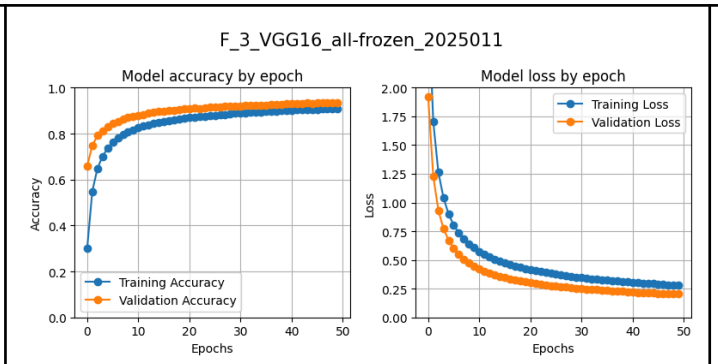
- **Test Accuracy**: 0.88
- **Validation Loss**: 0.20

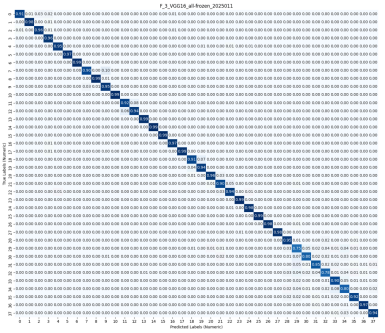
	Comparison		Variable	step time (ms/step)	Training		Validation		Test	Rank
					Accuracy	Loss	Accuracy	Loss	Accuracy	
1	Dataset size (2xConv2D, lr 0.0001)	A	20k	77	0.2955	2.4999	0.4235	2.3667	n.d	
		B	70k	55	0.4428	1.8830	0.6429	1.5910	n.d	
2-1	Image size_1 (2xConv2D, 70k, lr 0.00001)	C	224x224	55	0.9675	0.0955	0.9084	0.4316	0.79	
		J	256x256	70	0.9549	0.1352	0.8932	0.4974	0.73	
2-2	Image size_2 (3xConv2D, 70k, lr0.001)	G	224x224	68	0.9344	0.2313	0.9565	0.4183	0.76	
		H	256x256	86	0.9137	0.3149	0.9331	0.3099	0.85	
2-3	Image size_3 (3xConv2D, 70k, lr0.00001)	E	224x224	67	0.9977	0.0074	0.9530	0.2457	0.85	
		K	256x256	88	0.9965	0.0113	0.9554	0.2423	0.91	1
3-1	Learning rate (lr)_1 (2xConv2D, 70k, 224)	I	0.001	55	0.0276	3.6367	0.0286	3.6361	n.d	
		B	0.0001	55	0.4428	1.8830	0.6429	1.5910	n.d	
		C	0.00001	55	0.9675	0.0955	0.9084	0.4316	0.79	
3-2	Learning rate (lr)_2 (3xConv2D, 70k)	G	0.001	68	0.9344	0.2313	0.9565	0.4183	0.76	
		L	0.0001	67	0.9905	0.0397	0.9609	0.2867	0.82	
		E	0.00001	67	0.9977	0.0074	0.9530	0.2457	0.85	
4	Augmentation (2xConv2D, 70k, lr 0.00001)	C	No	55	0.9675	0.0955	0.9084	0.4316	0.79	
		D	Yes	335	0.8345	0.5464	0.8566	0.6177	0.85	3
5	CNN Layer (70k, lr 0.00001)	C	2x	55	0.9675	0.0955	0.9084	0.4316	0.79	
		E	3x	67	0.9977	0.0074	0.9530	0.2457	0.85	3
6	Model	E	CNN 3x (lr0.00001)	67	0.9977	0.0074	0.9530	0.2457	0.85	
		L	CNN 3x (lr0.0001)	67	0.9905	0.0397	0.9609	0.2867	0.82	
		F	VGG16 frozen (lr0.0001)	156	0.9076	0.2817	0.9360	0.2032	0.88	2

*BATCH_SIZE = 32, EPOCHS = 50
*IMG_SIZE = (224, 224) (except J, H), No augmentation(except C)
* VGG16: first 5xEpochs with lr 0.001, 50xEpochs with lr 0.00001, Dense layers 2x, Dropout 1x

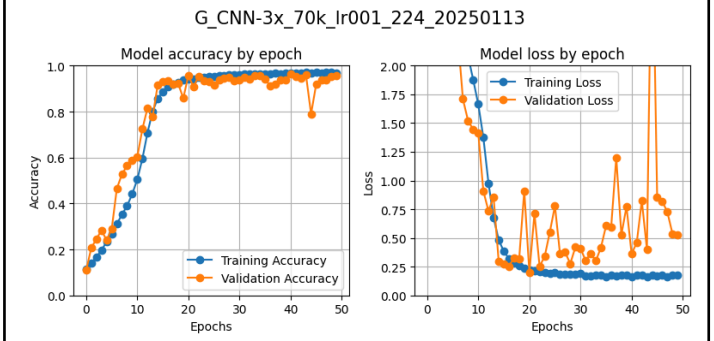
	Plot accuracies & loss over epochs	Confusion matrix (valid data)
A		
B		
C		
D		
E		

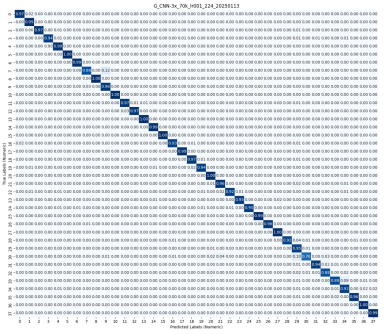
F


F_3_VGG16_all-frozen_2025011
Model accuracy by epoch
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Model loss by epoch
Loss
Epochs
Training Loss
Validation Loss

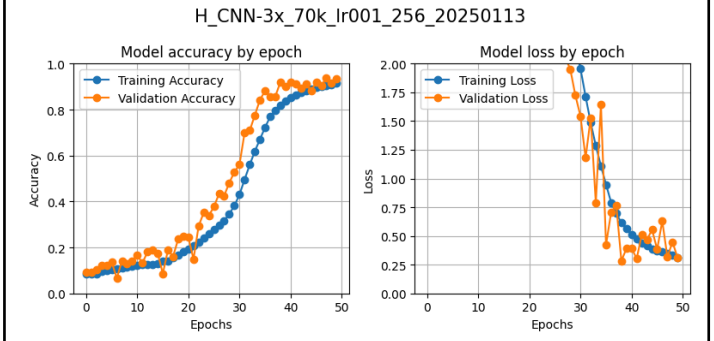


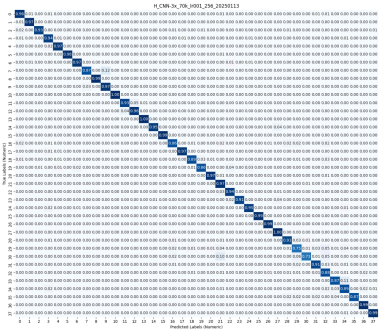
G


G_CNN-3x_70k_lr001_224_20250113
Model accuracy by epoch
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Model loss by epoch
Loss
Epochs
Training Loss
Validation Loss

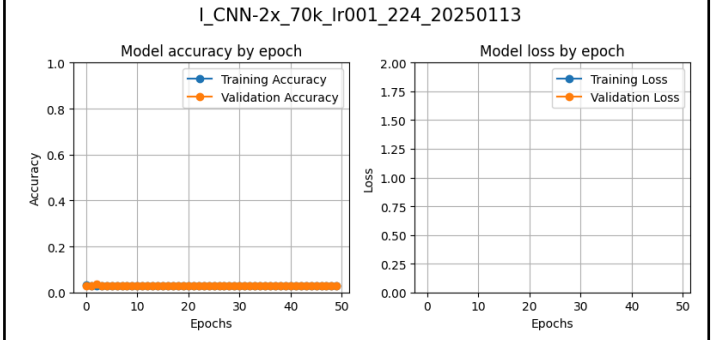


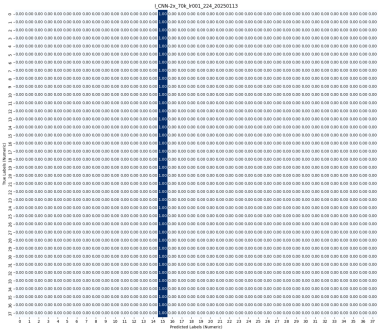
H


H_CNN-3x_70k_lr001_256_20250113
Model accuracy by epoch
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Model loss by epoch
Loss
Epochs
Training Loss
Validation Loss

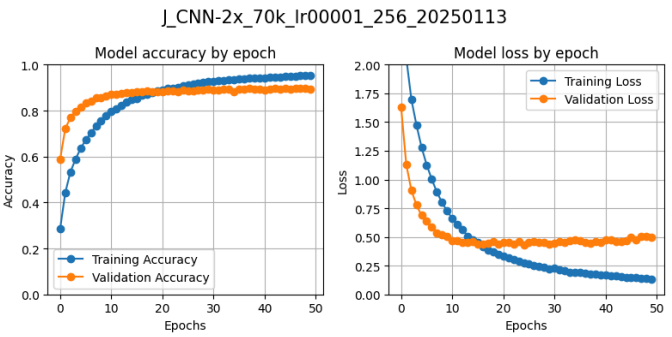


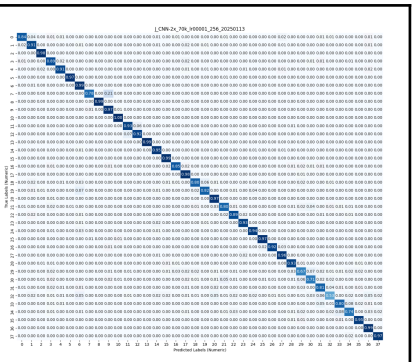
I


I_CNN-2x_70k_lr001_224_20250113
Model accuracy by epoch
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Model loss by epoch
Loss
Epochs
Training Loss
Validation Loss

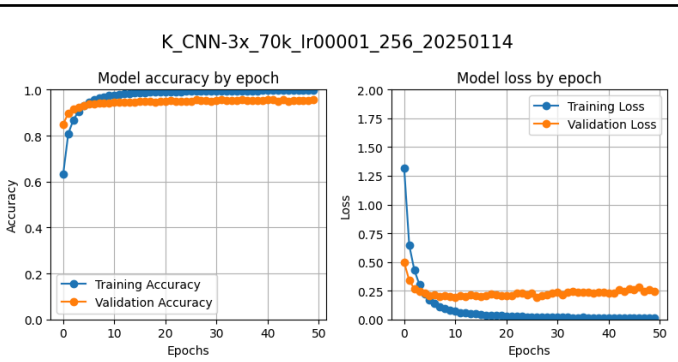


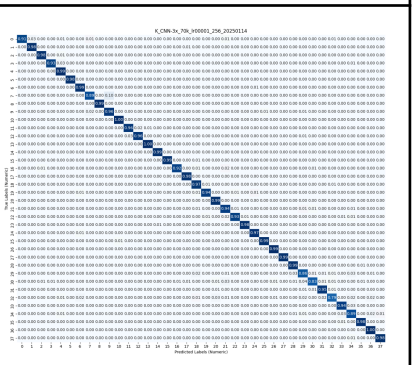
J


J_CNN-2x_70k_lr00001_256_20250113
Model accuracy by epoch
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Model loss by epoch
Loss
Epochs
Training Loss
Validation Loss

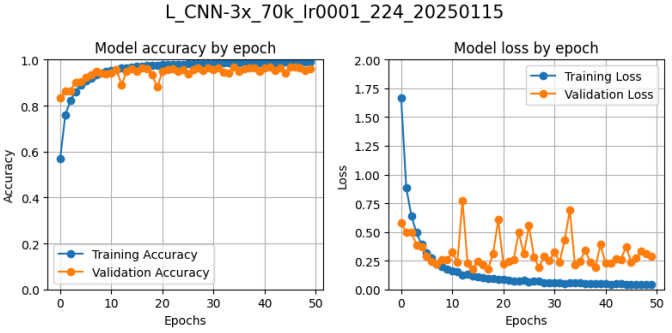


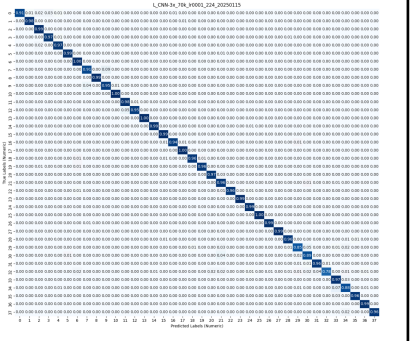
K


K_CNN-3x_70k_lr00001_256_20250114
Model accuracy by epoch
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Model loss by epoch
Loss
Epochs
Training Loss
Validation Loss

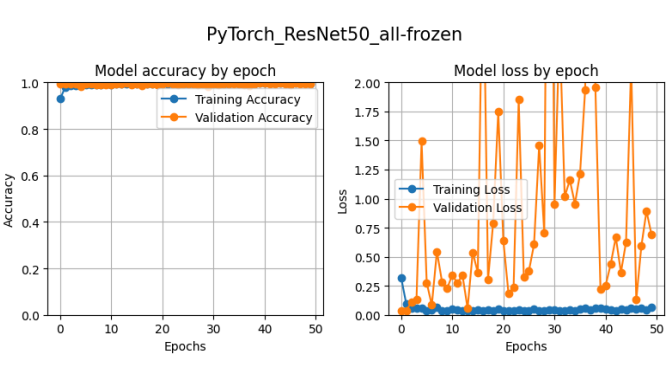


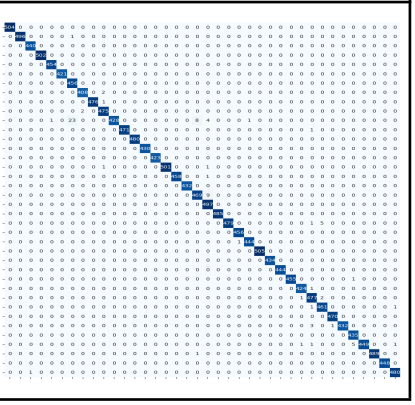
L


L_CNN-3x_70k_lr0001_224_20250115
Model accuracy by epoch
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Model loss by epoch
Loss
Epochs
Training Loss
Validation Loss



M

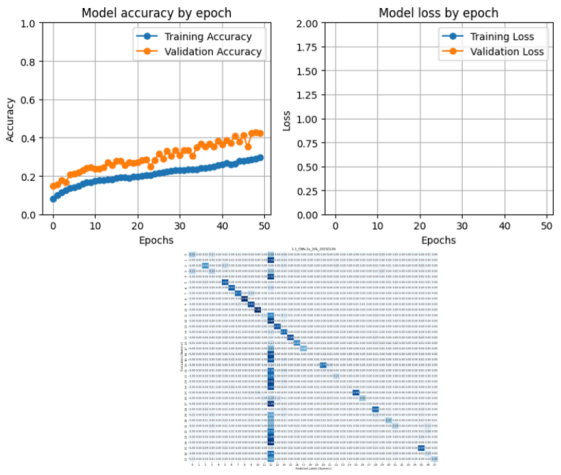

PyTorch_ResNet50_all-frozen
Model accuracy by epoch
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Model loss by epoch
Loss
Epochs
Training Loss
Validation Loss



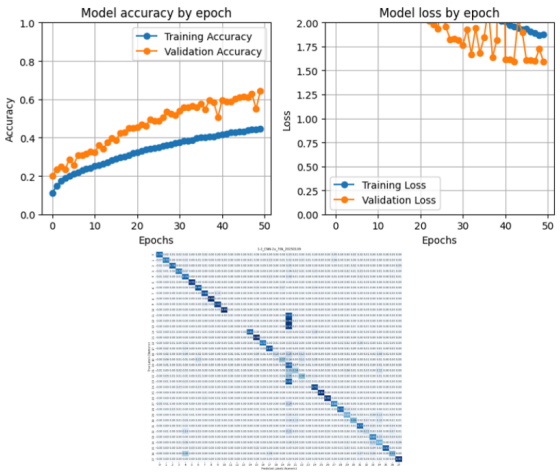
1. Size of dataset: 20k vs 70k
   - Basic model CNN 2x, lr 0.0001

	Comparison		Variable	step time (ms/step)	Training		Validation		Test	Rank
					Accuracy	Loss	Accuracy	Loss	Accuracy	
1	Dataset size (2xConv2D, lr 0.0001)	A	20k	77	0.3	2.5	0.42	2.37	n.d	2
		B	70k	55	0.44	1.88	0.64	1.59	n.d	1

A_CNN-2x_20k



B_CNN-2x_70k



* Model: CNN 2x, lr0.0001, no augmentation

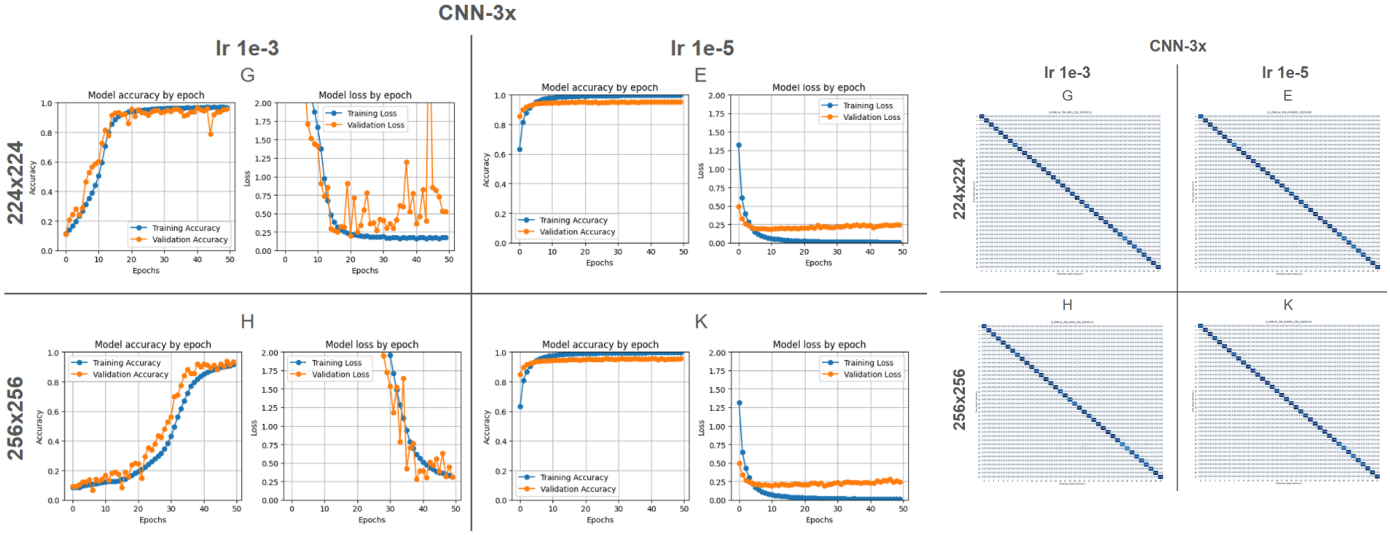
- **Dataset Size Impact**: The 70k dataset significantly improves training and validation accuracy (0.44/0.64) compared to the 20k dataset (0.3/0.42), reducing validation loss (1.59 vs. 2.37) and enhancing generalization.
- **Confusion Matrix**: The 70k dataset has stronger diagonal dominance, reflecting better classification performance with fewer misclassifications compared to the 20k dataset

Insight:

- A larger dataset (70k) significantly boosts the model's performance compared to a smaller dataset (20k).

2. Image size: 224x224 vs 256x256

	Comparison		Variable	step time (ms/step)	Training		Validation		Test	Rank
					Accuracy	Loss	Accuracy	Loss	Accuracy	
2-2	Image size_2 (3xConv2D, 70k, lr 1e-3)	G	224x224	68	0.93	0.23	0.95	0.41	0.76	4
		H	256x256	86	0.91	0.31	0.93	0.31	0.85	3
2-3	Image size_3 (3xConv2D, 70k, lr 1e-5)	E	224x224	67	1.00	0.01	0.95	0.25	0.85	2
		K	256x256	88	1.00	0.01	0.96	0.24	0.91	1



- With a higher learning rate (lr = 1e-3), larger image sizes (256x256, Variable H) demonstrate a more stable and gradual increase in accuracy compared to smaller image sizes (224x224, Variable G). This indicates that larger images help stabilize the learning process.
- Increasing the image size from 224x224 to 256x256 improves the test accuracy for both learning rates.
- The smoother loss curve and gradual accuracy improvement for larger image sizes (256x256) at lr = 1e-3 suggest better generalization and consistent learning, making it a more reliable choice despite slower accuracy gains.

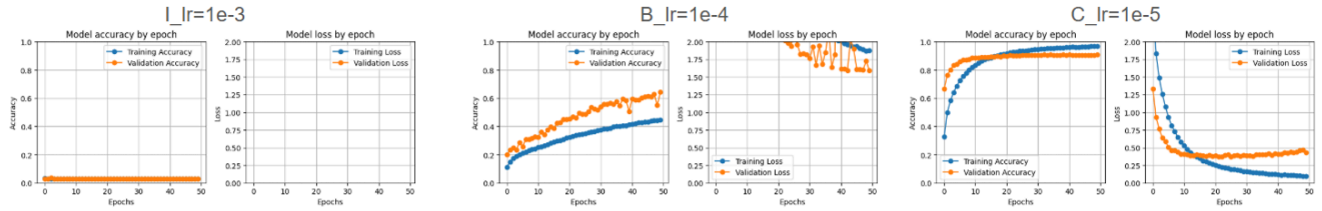
Insight: Larger image size (256x256) improves model performance.

3. Learning rate (lr): 1e-3 vs 1e-4 vs 1e-5

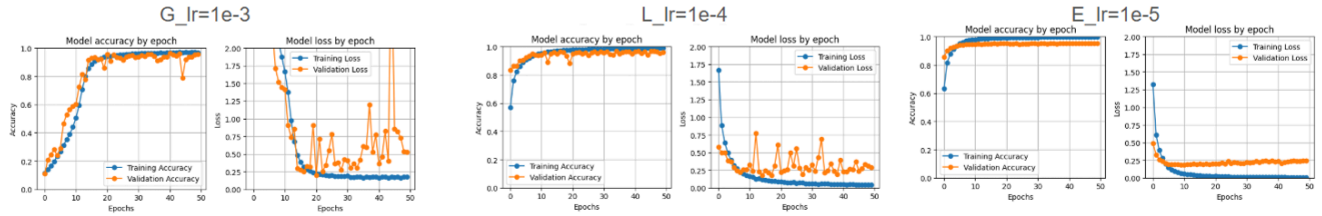
- Basic model CNN_70k

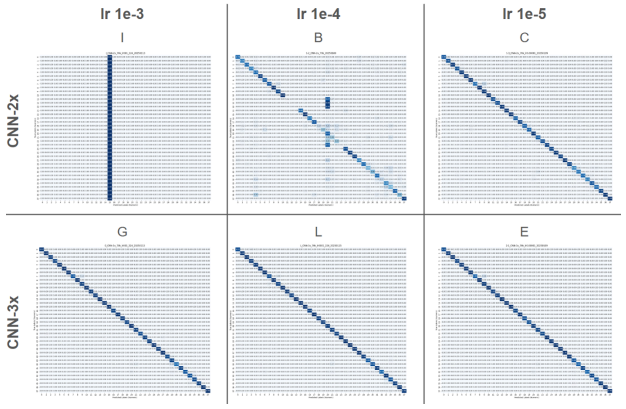
	Comparison		Variable	step time (ms/step)	Training		Validation		Test	Rank
					Accuracy	Loss	Accuracy	Loss	Accuracy	
3-1	Learning rate (lr)_1 (2xConv2D, 70k, 224)	I	1e-3	55	0.0276	3.6367	0.0286	3.6361	n.d	
		B	1e-4	55	0.4428	1.8830	0.6429	1.5910	n.d	
		C	1e-5	55	0.9675	0.0955	0.9084	0.4316	0.79	3
3-2	Learning rate (lr)_2 (3xConv2D, 70k, 224)	G	1e-3	68	0.9344	0.2313	0.9565	0.4183	0.76	4
		L	1e-4	67	0.9905	0.0397	0.9609	0.2867	0.82	2
		E	1e-5	67	0.9977	0.0074	0.9530	0.2457	0.85	1

1: CNN-2x



2: CNN-3x





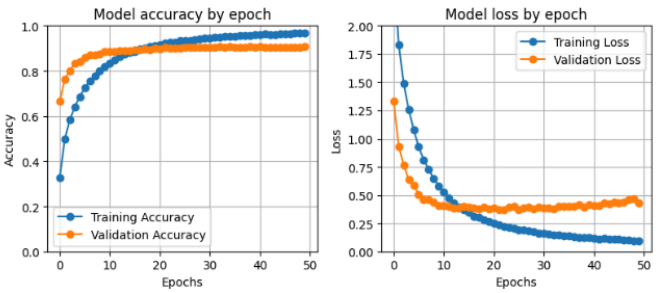
- For both architectures, a learning rate of 1e−5 (Models C and E) results in the best performance.
- Lowering the learning rate significantly improves classification accuracy by enhancing diagonal dominance in the confusion matrix, and deeper architectures (CNN-3x) further amplify this improvement.

Insight: Lower learning rates (1e-5) combined with deeper architectures (CNN-3x) achieve better model performance.

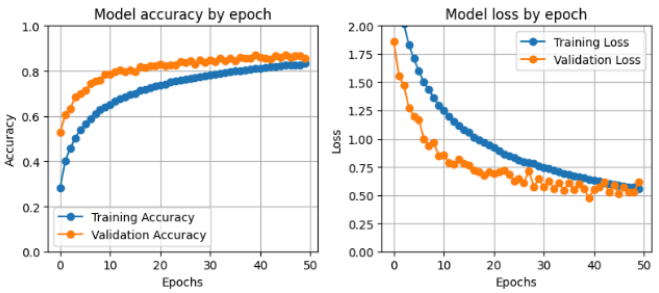
4. Augmentation: No vs Yes

	Comparison		Variable	step time (ms/step)	Training		Validation		Test	Rank
					Accuracy	Loss	Accuracy	Loss	Accuracy	
4	Augmentation (2xConv2D, 70k, lr 1e-5)	C	No	55	0.96	0.1	0.91	0.43	0.79	2
		D	Yes	335	0.83	0.55	0.86	0.62	0.85	1

C_No Augmentation

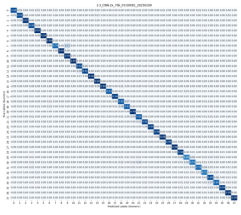


D_Augmentation *

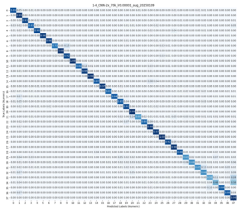


* Augmentation: shearing, zoom, horizontal flip
** Model: CNN 2x, 70k, lr0.00001

C_No Augmentat



D_Augmentation



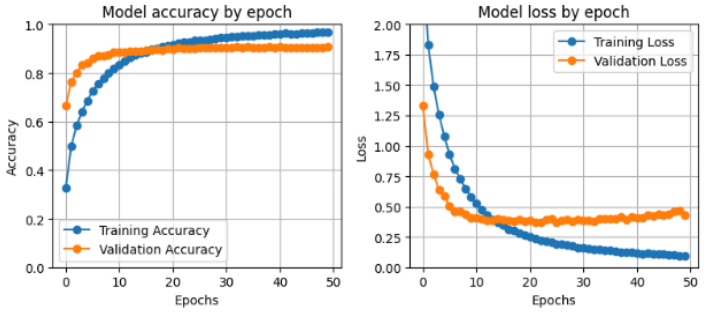
- Without augmentation (Model C), training accuracy is higher, but it seems there is overfitting, as the validation accuracy is slightly lower than the training accuracy.
- With augmentation (Model D), training and validation accuracy are lower, and loss is higher, but the model generalizes better, as shown by improved test accuracy (0.85 vs. 0.79).
- Data augmentation increases training time (335 ms/step vs. 55 ms/step)

Insight: Data augmentation enhances model generalization, by reducing overfitting but significantly increases computational costs.

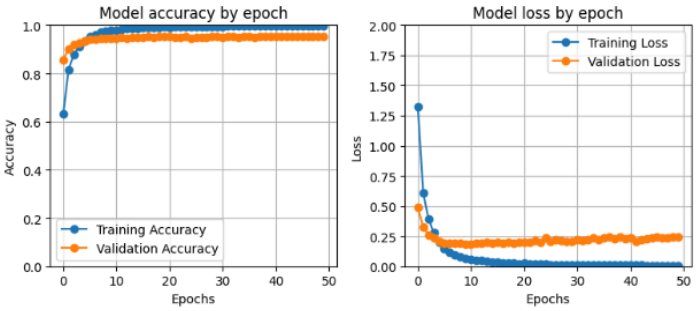
5. CNN layer: 2x vs 3x

	Comparison		Variable	step time (ms/step)	Training		Validation		Test	Rank
					Accuracy	Loss	Accuracy	Loss	Accuracy	
5	CNN Layer (70k, lr 1e-5)	C	2x	55	0.96	0.1	0.91	0.43	0.79	2
		E	3x	67	1.00	0.01	0.95	0.25	0.85	1

C_2x layers

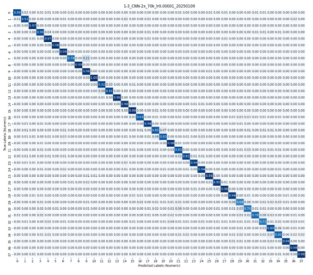


E_3x layers

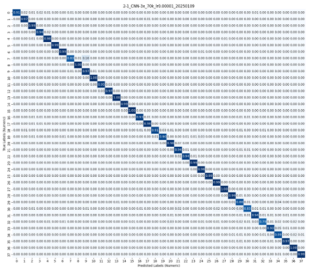


* Model: 70k, lr0.00001, no augmentation

C_2x layers



E_3x layers

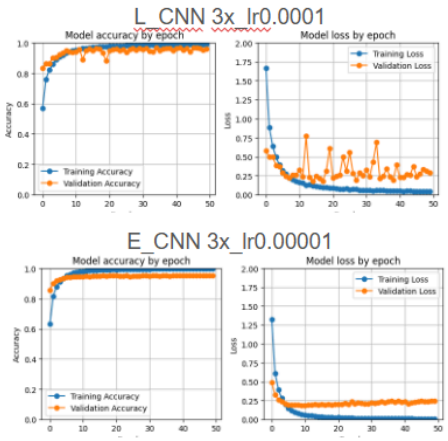


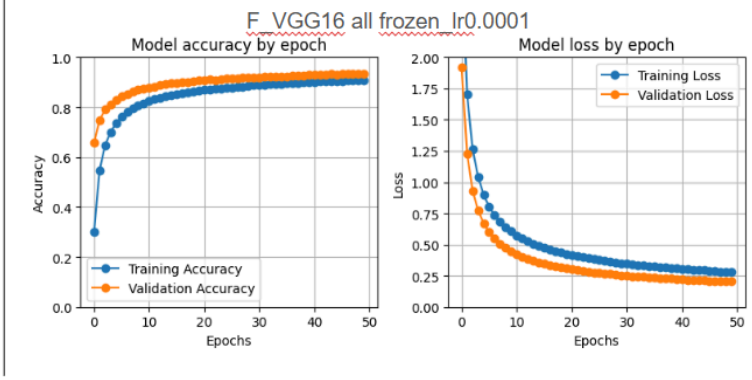
- Adding more layers (3x compared to 2x) enhances the model's ability to learn and generalize, as reflected in higher validation accuracy and lower validation loss.
- The trade-off is increased computational cost (longer step time) and potential overfitting, as the 3x layers model achieves perfect training accuracy.
- Test accuracy (C:0.79 vs E: 0.85) confirms that the 3x layers model generalizes better to unseen data.

Insight: Deeper architecture (3x layers) improves generalization and test accuracy (0.85 vs. 0.79) at a modest computational cost, making it a better choice for complex classification tasks.

6. Model: CNN 3x VGG16 all frozen

	Comparison		Variable	step time (ms/step)	Training		Validation		Test
					Accuracy	Loss	Accuracy	Loss	Accuracy
6	Model	E	CNN 3x (lr0.00001)	67	0.9977	0.0074	0.9530	0.2457	0.85
		L	CNN 3x (lr0.0001)	67	0.9905	0.0397	0.9609	0.2867	0.82
		F	VGG16 frozen (lr0.0001)	156	0.9076	0.2817	0.9360	0.2032	0.88





* CNN 3x: 224, 70k, no augmentation
** VGG16: 224, 70k, no augmentation

- The difference between CNN and VGG16 is better highlighted by comparing CNN model L, which uses the same learning rate as VGG16.
- VGG16 has smoother and more stable learning curves for both accuracy and loss, indicating a **slower and steadier learning process** compared to the CNN models.
- VGG's higher test accuracy confirms VGG16's pre-trained features help it generalize better, especially with unseen test data, despite its frozen layers.
- The VGG16 model requires significantly more computation time per step due to its larger and more complex architecture.

Insight:

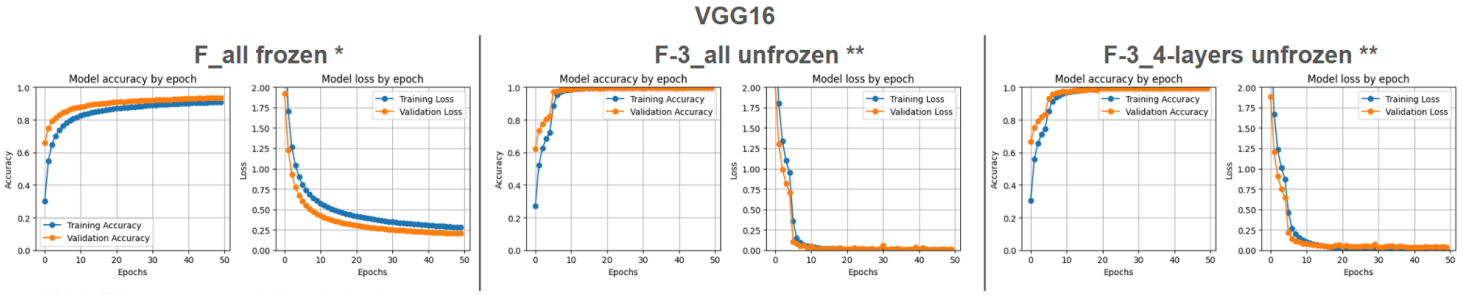
- The CNN 3x model achieves perfect training accuracy, whereas the frozen VGG16 model shows lower training accuracy and higher training loss. This is expected since VGG16's layers are frozen and not updated during training, limiting its capacity to adapt to our dataset.
- 70k dataset is small for high-complexity tasks (38 classes). Therefore pre-trained models such as VGG16 would perform better because it leverages features learned from a large dataset like ImageNet(~14M images). This allows it to generalize better with small training samples.

- I would be better fine-tunning pre-trained models, rather than training CNNs from scratch.

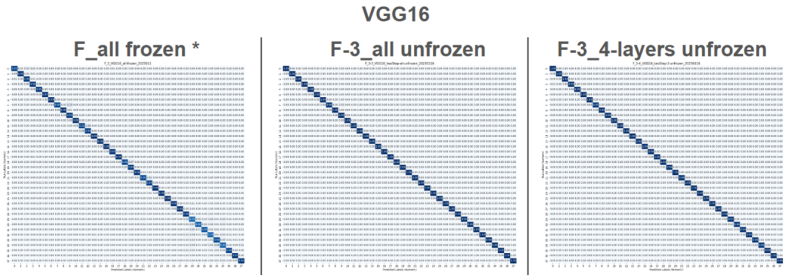
7. Fine tuning VGG16

	Comparison		Variable	step time (ms/step)	Training		Validation		Test	Rank
					Accuracy	Loss	Accuracy	Loss	Accuracy	
1	VGG16	F-1	All frozen	156	0.9095	0.2796	0.9346	0.2046	0.88	3
		F-3	unfrozen-All	68	0.9993	0.0030	0.997	0.0089	1.00	1
		F-4	unfrozen-4 layers	52	0.9956	0.0144	0.9874	0.0389	0.97	2

* IMG_SIZE = (224, 224) , No augmentation
* VGG16: total 50x Epochs, Dense layers 2x, Dropout 1x
* All frozen: lr 1e-4 / All & 4-layers unfrozen 1e-4/1e-5



* 224, 70k, no augmentation, lr 1e-4
** 224, 70k, no augmentation, lr 1e-4/1e-5



To enhance the performance of VGG16, several key modifications were made:

1. **Image Size Adjustment**
   The input image size was set to 224 by 224 pixels, which matches the dimensions of the pretrained ImageNet dataset. This adjustment enabled better alignment between our dataset and the pretrained weights, significantly improving the model's performance.

2. **Learning Rate Optimization**
   A lower learning rate was used, which proved to be an essential factor in stabilizing the training process and improving accuracy.

3. **Progressive Fine-Tuning Approach**
   We adopted a progressive fine-tuning strategy to train the model effectively. This approach involved the following steps:

- **Step 1:** The model was trained for 5 epochs with all layers frozen, allowing the newly added layers to adapt to the dataset without disrupting the pretrained feature extractor.
- **Step 2:** All layers (or the last block) were unfrozen, and the model was trained for an additional 45 epochs using a smaller learning rate. This step fine-tuned the higher-level features to align better with our dataset's specific requirements.

The VGG16 architecture, with all layers unfrozen, demonstrated the best performance, achieving a high level of prediction accuracy.

Dataset Expansion

Initially, our test dataset consisted of 33 images across 8 classes, which was insufficient for robust evaluation. To address this, we expanded the dataset by sourcing additional plant leaf images from the PlantPAD website. This expansion included more images and added 3 new classes, significantly enhancing the diversity of the test dataset.

Test dataset: 33 images		
Class		No.
1	Apple cedar rust	4
2	Apple scab	3
3	Corn common rust	3
4	Potato early blight	5
5	Potato healty	2
6	Tomato early blight	6
7	Tomato healthy	4
8	Tomato yellow curl virus	6

Test dataset: 283 images		
Class		No.
1	Apple cedar rust	4
2	Apple scab	3 + 50
3	Corn common rust	3
4	Potato early blight	5
5	Potato healty	2
6	Tomato early blight	6
7	Tomato healthy	4
8	Tomato yellow curl virus	6 + 50
9	Strawberry leaf scorch	50
10	Tomato mosaic virus	50
11	Tomato target spot	50

F-3 VGG16 unfrozen-All	Test accuracy	
	33 images	283 images
	100 %	100 %

The best-performing model, VGG16, achieved excellent accuracy on the expanded dataset. This demonstrates that the model was not only capable of learning from the training data but also generalized well to unseen data.