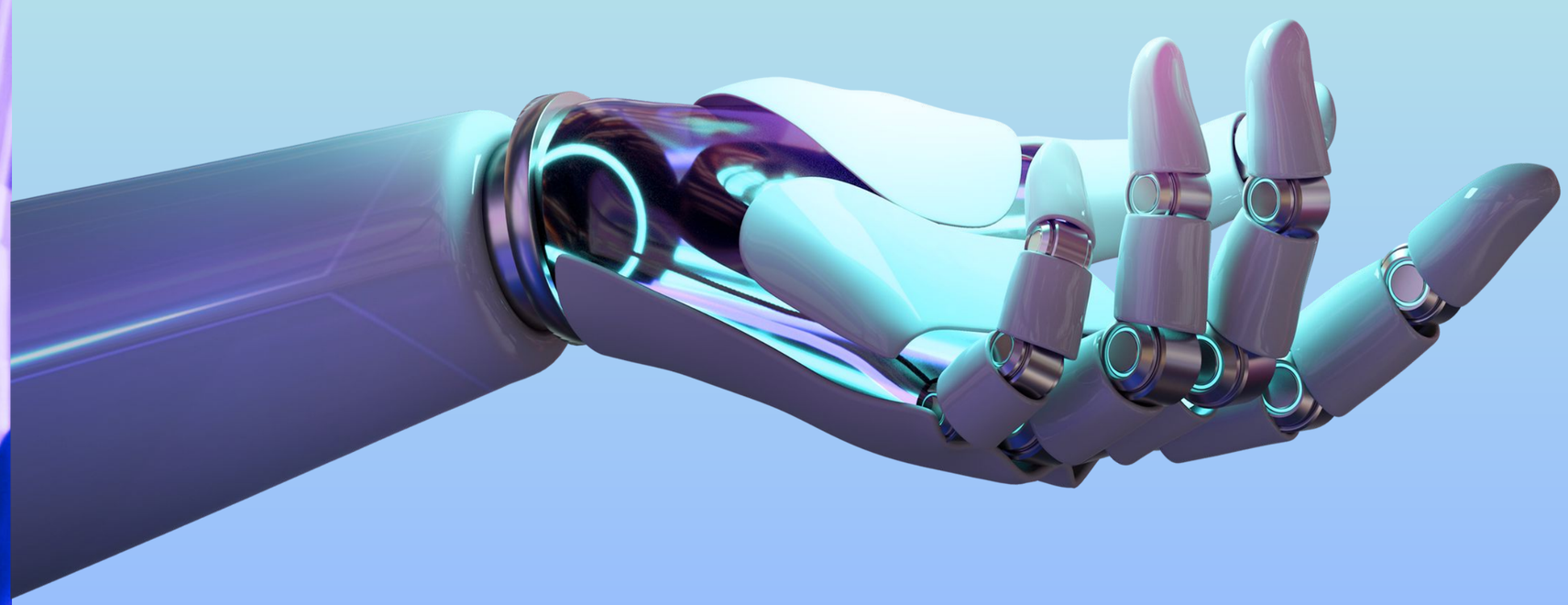
Image Recognition using CIFAR-10

Comparison of CNN architectures and transfer-learning methods on CIFAR-10



Project by:
Bosco,
Gabriele,
Mara

aka

The Fantastic Three

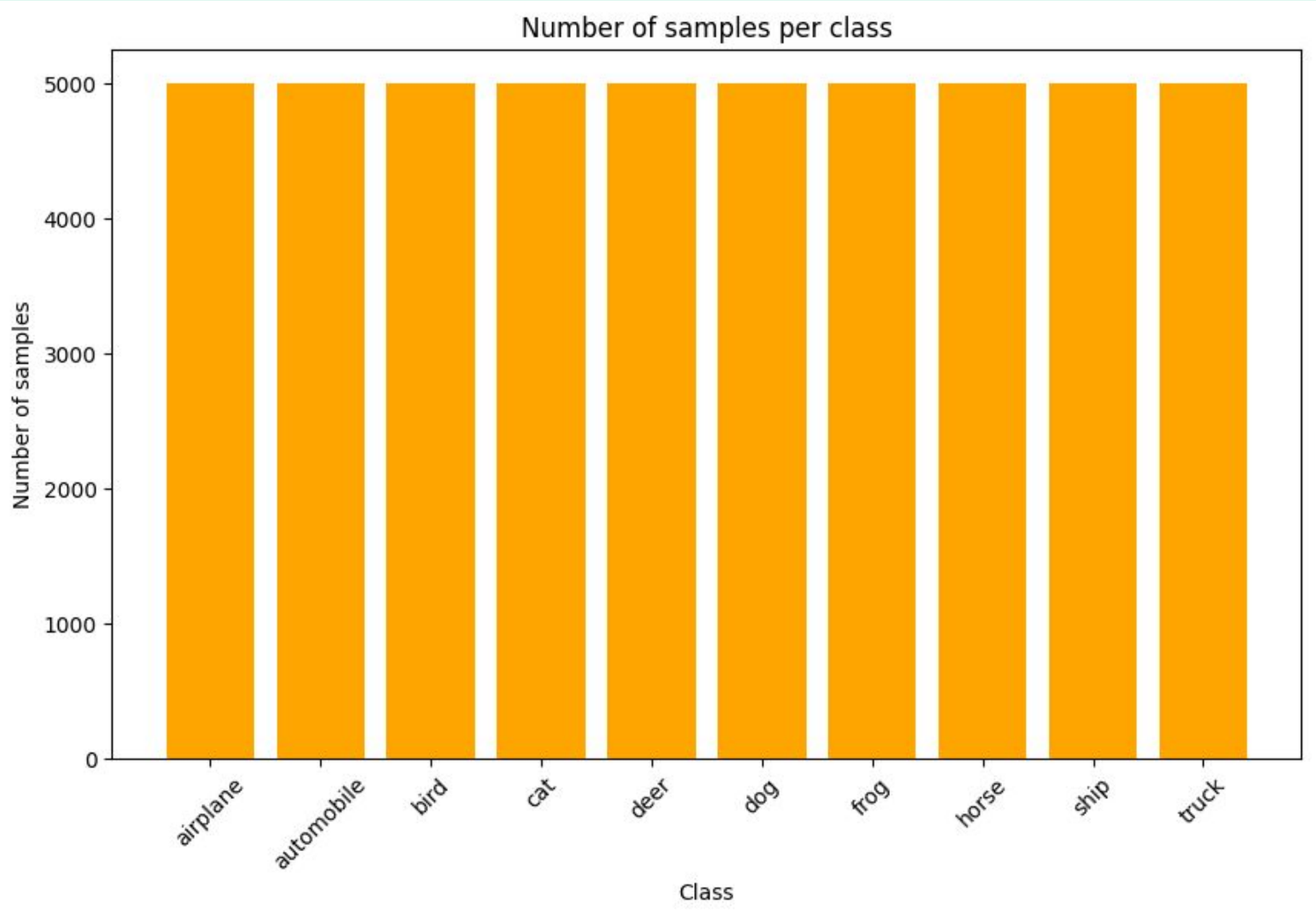
Project Goal

- Production of a robust classifier
- Comparison baseline CNNs vs transfer learning
- Fine tuning through layer & parameter control

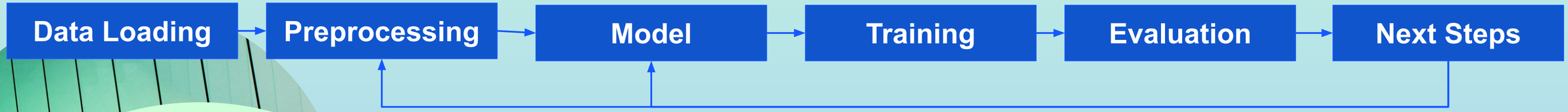
Dataset CIFAR-10

- 60 images, 10 classes, 32×32 RGB
- Train/test split: 50k / 10k
- Evenly distributed classes

Label	Description
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck



Project Workflow



Key Hyperparameters:

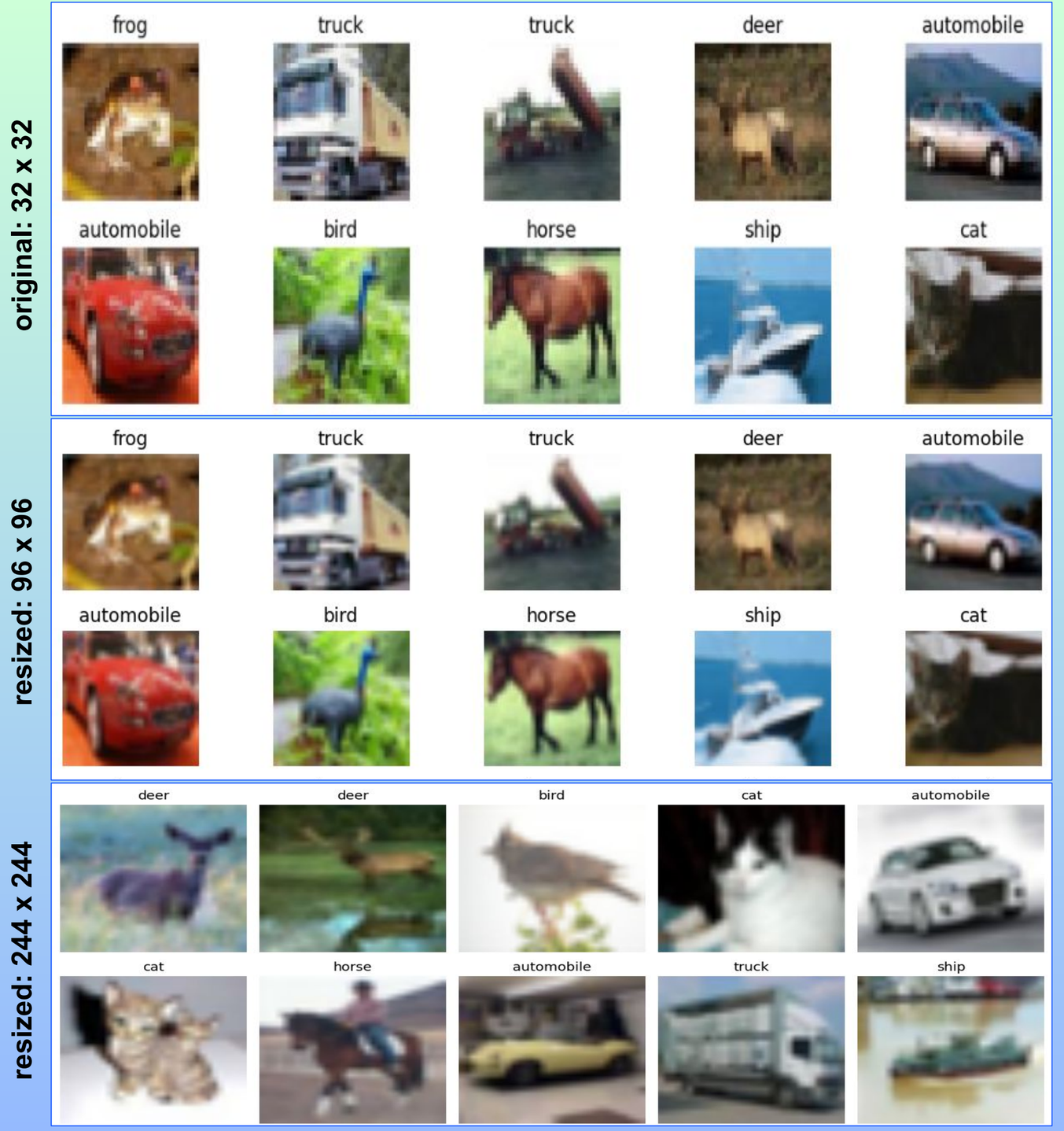
- Epochs, Batch Size, Optimizer, Early Stopping

Addition:

- reproducibility: random seeds
- validation schema (validation split vs test)
- early stopping to avoid overfitting

Preprocessing

- Normalization (scale to [0 , 1])
- One-hot encoding of labels
- Resizing images for transfer learning:
  - 96 x 96 for MobilNetV2
  - 224 x 224 for DenseNet121
  - 224 x 224 for EfficientNetB0
  - 224 x 224 for NASNetMobile
- Data augmentation



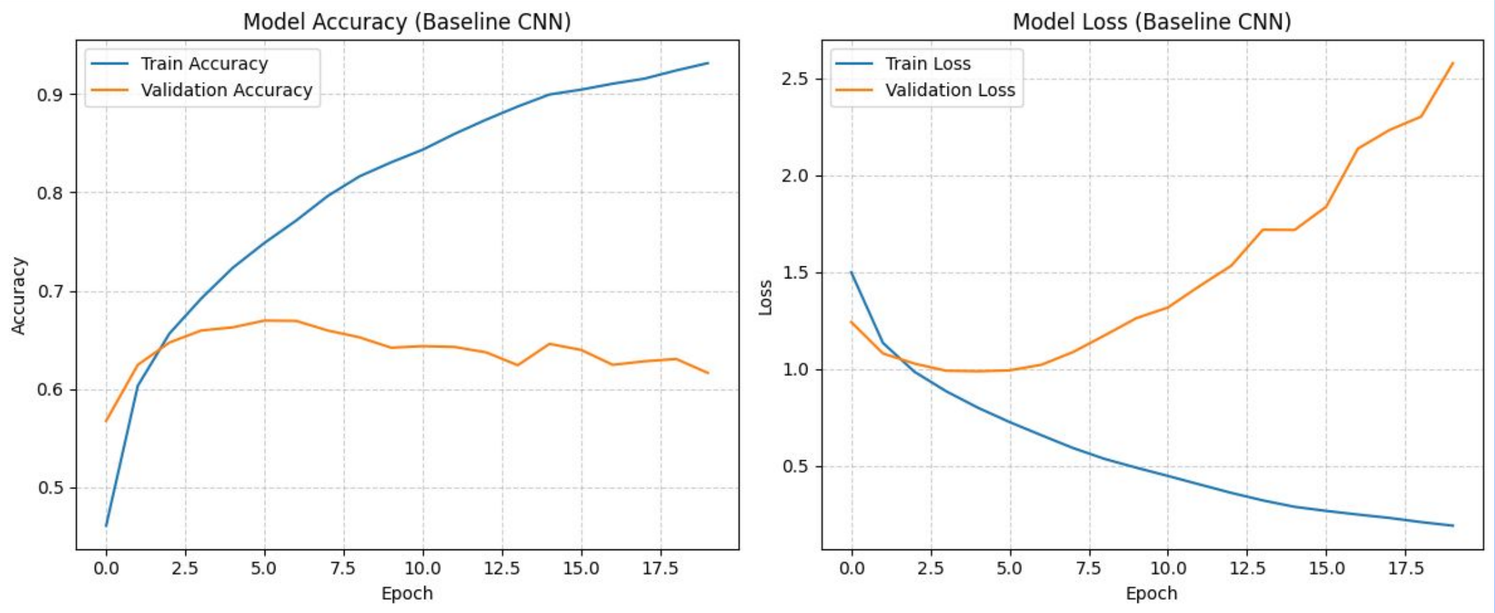
Baseline CNN Architecture

CNN Baseline Architecture:

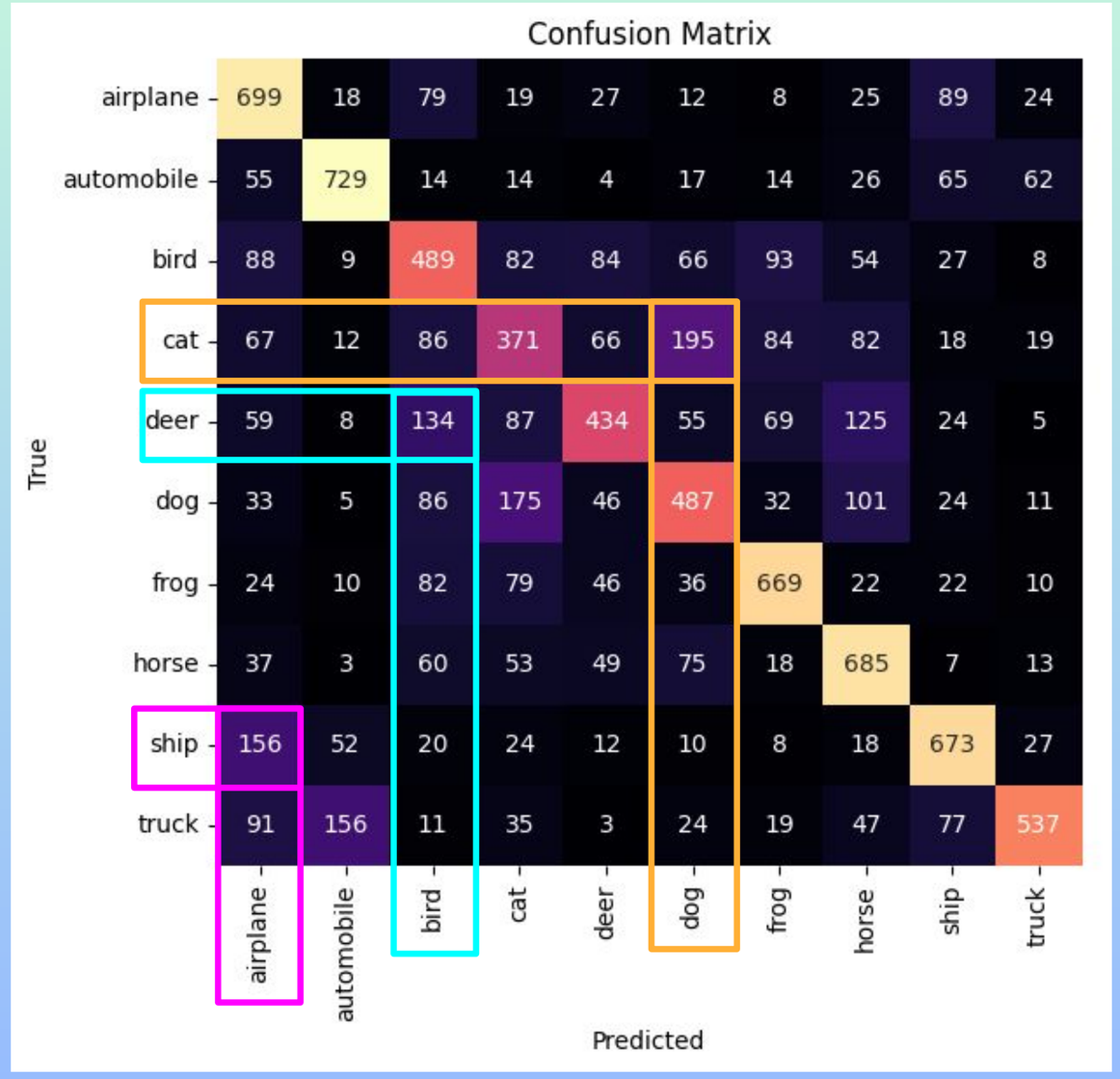
```
model_base = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

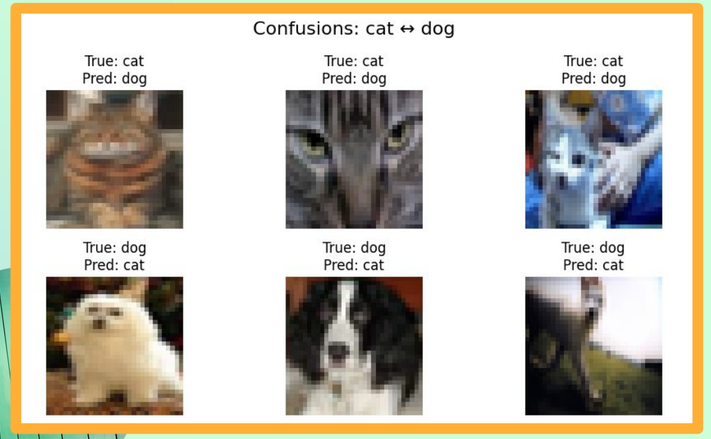
CNN Baseline Training:

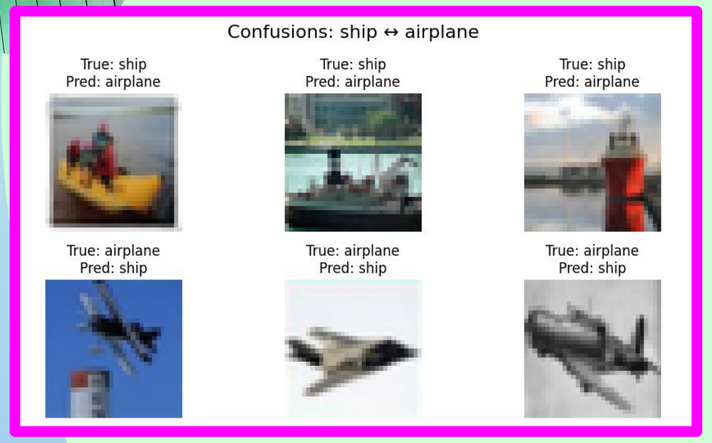
```
model_base.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
history_base = model_base.fit(X_train, y_train_cat,
epochs=20, validation_split=0.2, shuffle=True)
```

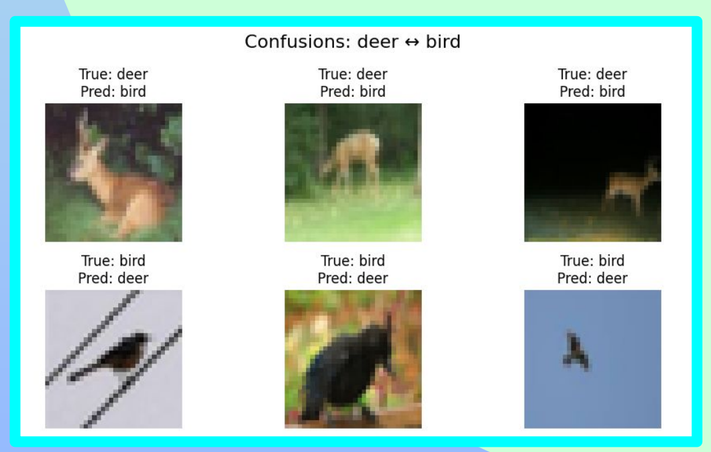


Baseline CNN



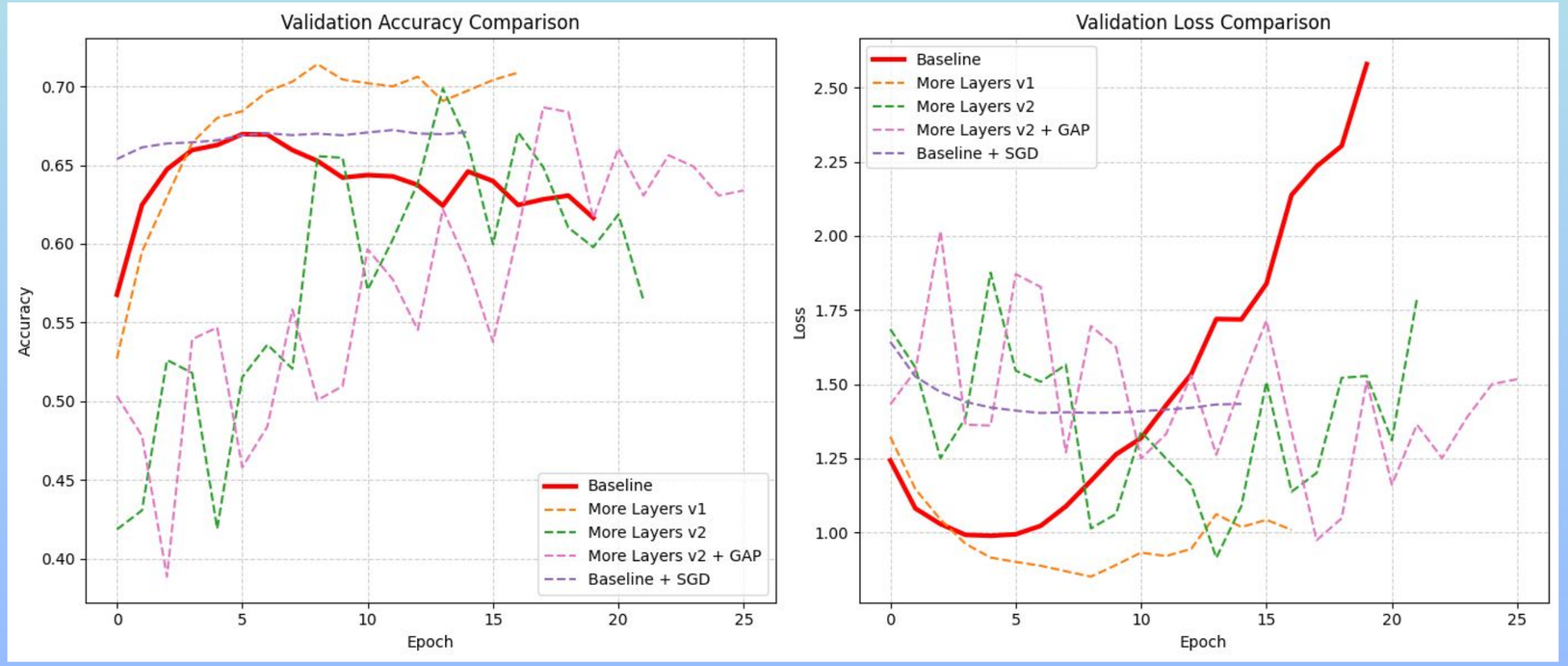






Stronger CNN designs & variants

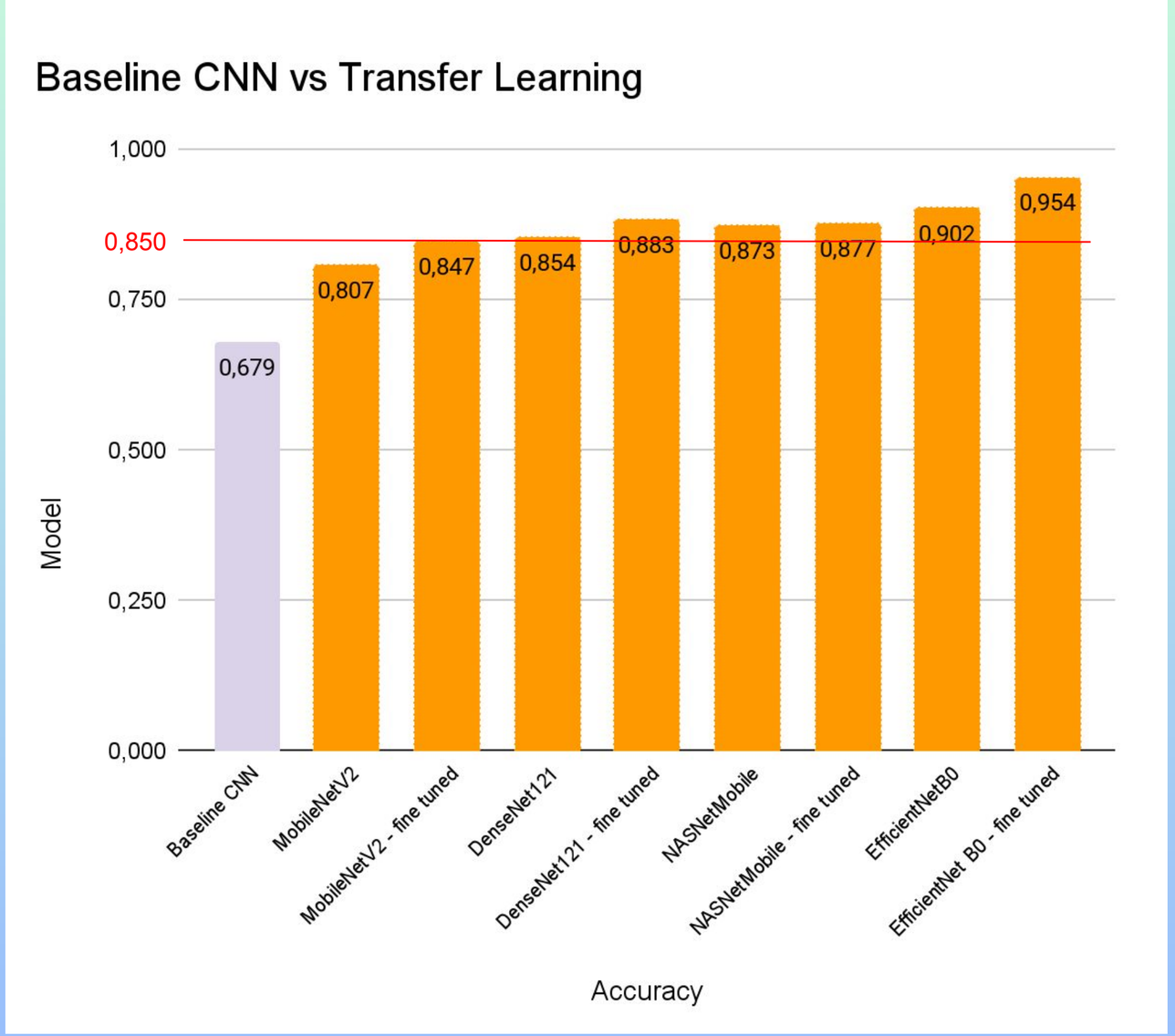
Trial	Model	Key Addition	Accuracy
1	Baseline CNN	2 Conv layers + Dense	0.679
2	CNN More Layers V1	+1 Conv layer, Dropout	0.711
3	CNN More Layers V2	BatchNorm, padding=”same”	0.685
4	CNN More Layers V2 + GAP	+ GAP	0.678
5	CNN Optimizer SGD	Optimizer change	0.651



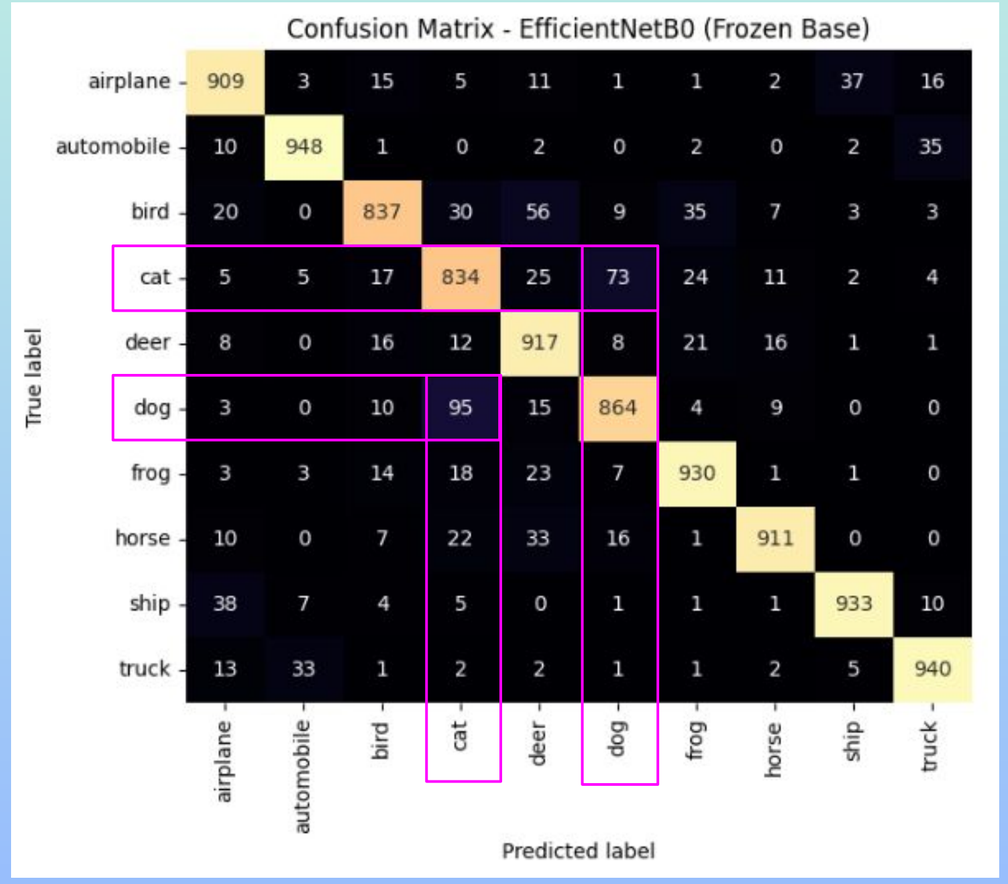
Transfer Learning

Comparison of 4 different ImageNet pretrained models:

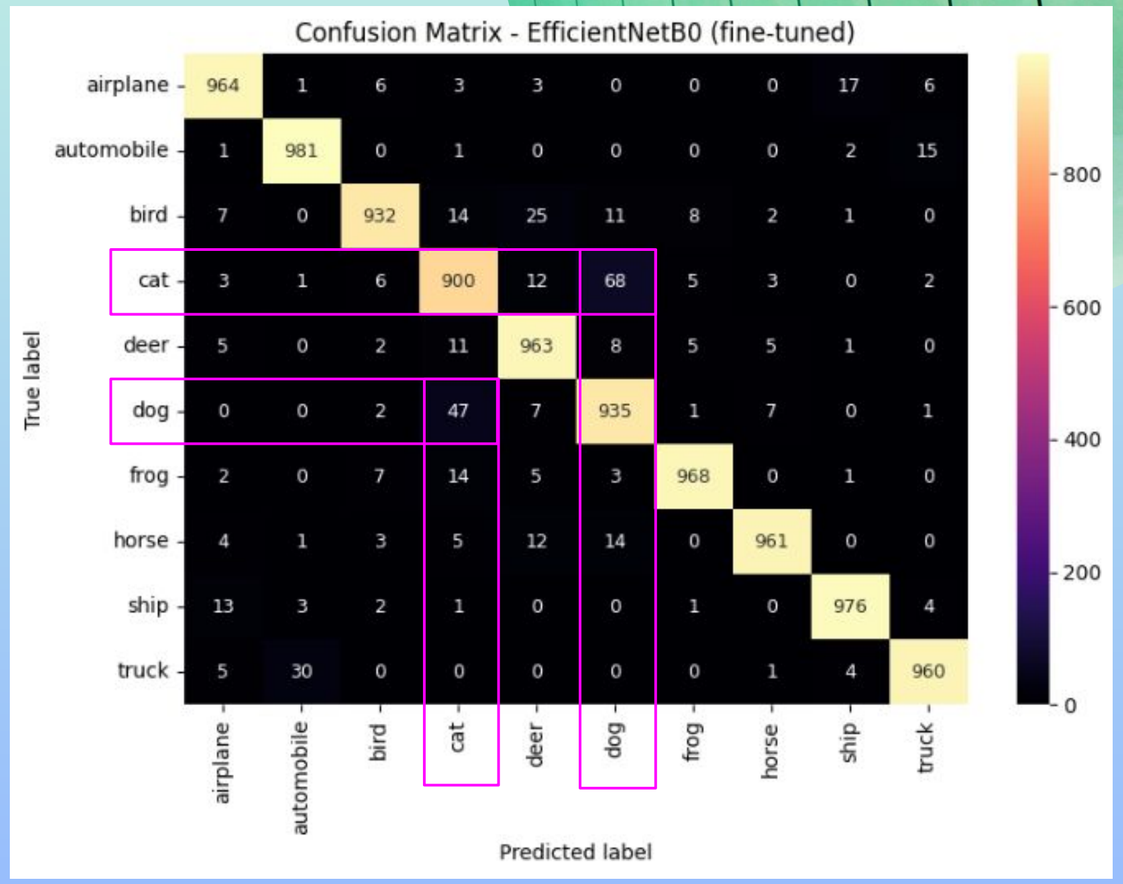
- Comparison accuracy to baseline CNN model
- Comparison fixed pretrained parameters to fine-tune



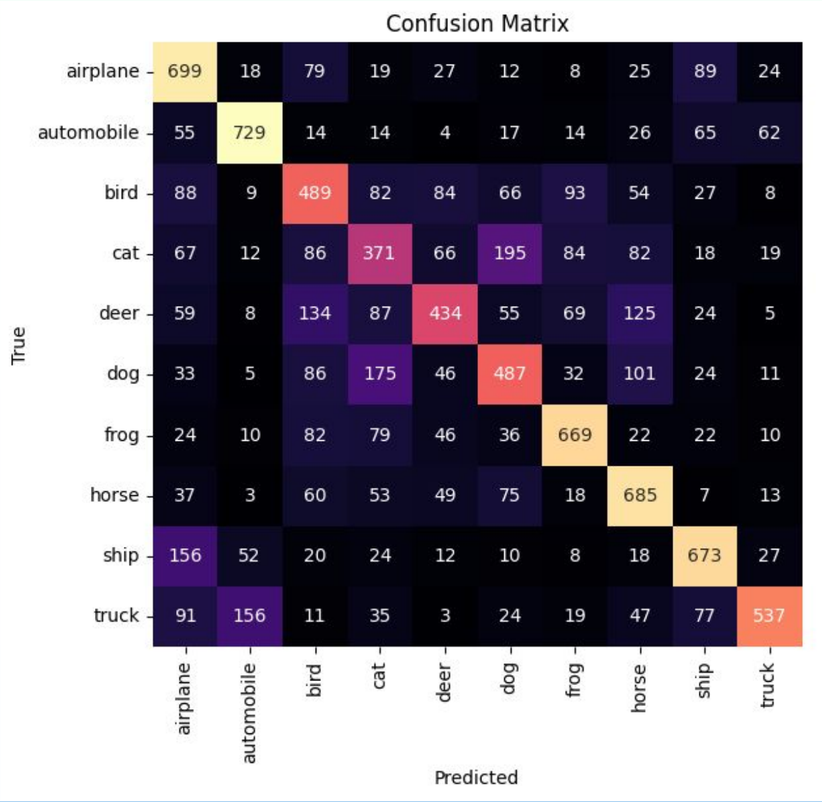
Transfer Learning - Best Model (EfficientNetB0)

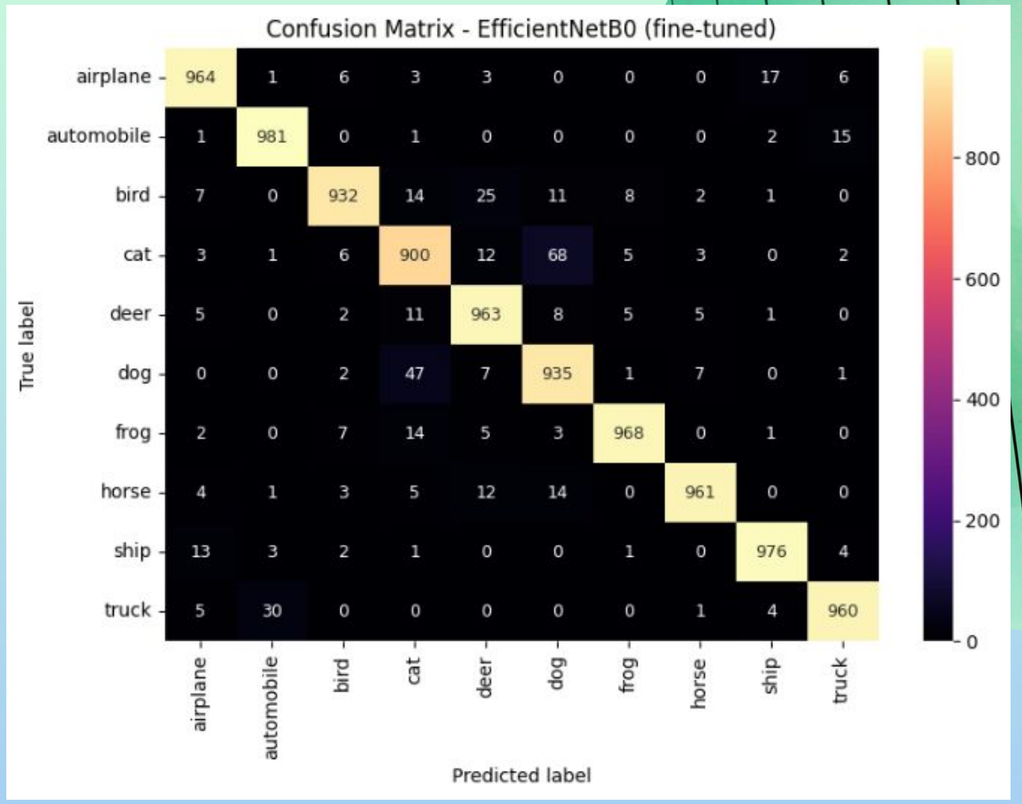


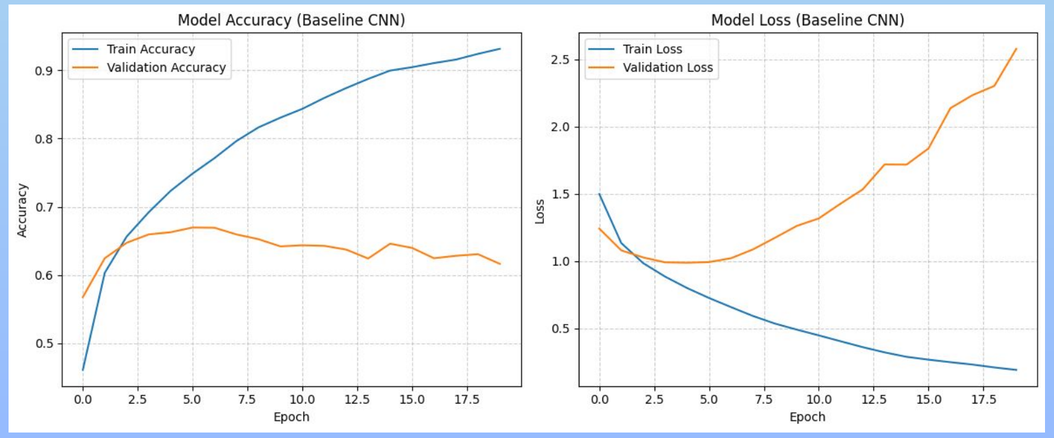
unfreeze last 50/132 layers (38%)

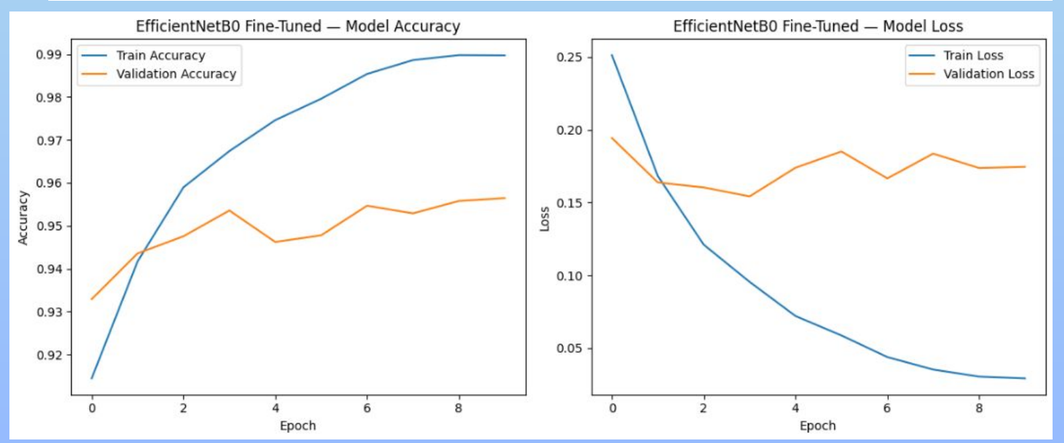


CNN vs Transfer Learning









Next Steps

Challenge: slight overfit and/or too homogeneous data

EfficientNetB0 model:

- further fine tuning
  - unfreezing up to 70 layers (~50%)
  - smaller learning rate: from 0.001 (default) to 0.0001 or even 0.0001

    → can lead to alignment of validation accuracy & loss to training curve

- evaluation on robustness
  - testing augmented (more noisy) samples

Augmentation – Outlook

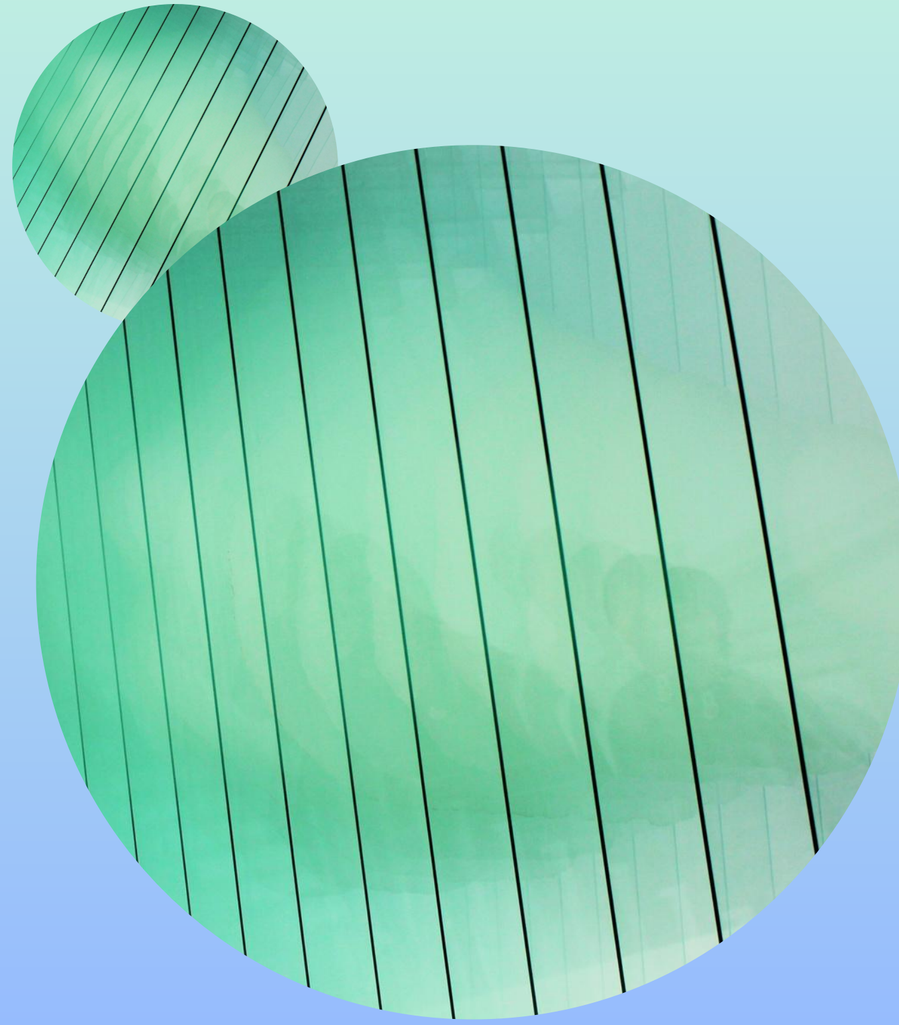
```
Epoch 1/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 160s 152ms/step - accuracy: 0.8279 - loss: 0.5223 - val_accuracy: 0.8581 - val_loss: 0.4093
Epoch 2/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 75s 95ms/step - accuracy: 0.8978 - loss: 0.2923 - val_accuracy: 0.8830 - val_loss: 0.3616
Epoch 3/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 74s 95ms/step - accuracy: 0.9235 - loss: 0.2119 - val_accuracy: 0.8803 - val_loss: 0.3877
Epoch 4/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 74s 95ms/step - accuracy: 0.9447 - loss: 0.1546 - val_accuracy: 0.8841 - val_loss: 0.3920
Epoch 5/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 74s 95ms/step - accuracy: 0.9552 - loss: 0.1286 - val_accuracy: 0.8864 - val_loss: 0.4293
Epoch 6/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 74s 95ms/step - accuracy: 0.9620 - loss: 0.1043 - val_accuracy: 0.8894 - val_loss: 0.4070
Epoch 7/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 74s 95ms/step - accuracy: 0.9691 - loss: 0.0856 - val_accuracy: 0.8884 - val_loss: 0.4638
Epoch 8/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 74s 95ms/step - accuracy: 0.9725 - loss: 0.0737 - val_accuracy: 0.8886 - val_loss: 0.5409
Epoch 9/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 74s 95ms/step - accuracy: 0.9765 - loss: 0.0661 - val_accuracy: 0.8865 - val_loss: 0.5022
Epoch 10/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 74s 95ms/step - accuracy: 0.9808 - loss: 0.0576 - val_accuracy: 0.8900 - val_loss: 0.5122
DenseNet121 FineTuned -> Test Accuracy: 0.8830, Loss: 0.3616
```

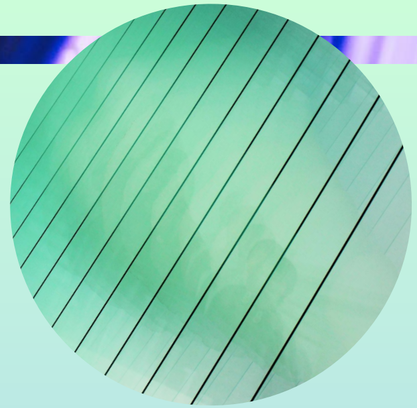
```
Epoch 1/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 697s 840ms/step - accuracy: 0.2318 - loss: 2.6198 - val_accuracy: 0.6188 - val_loss: 1.1167
Epoch 2/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 620s 793ms/step - accuracy: 0.5147 - loss: 1.4582 - val_accuracy: 0.7140 - val_loss: 0.8367
Epoch 3/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 625s 799ms/step - accuracy: 0.6137 - loss: 1.1602 - val_accuracy: 0.7567 - val_loss: 0.7147
Epoch 4/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 624s 798ms/step - accuracy: 0.6547 - loss: 1.0274 - val_accuracy: 0.7812 - val_loss: 0.6402
Epoch 5/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 624s 798ms/step - accuracy: 0.6882 - loss: 0.9177 - val_accuracy: 0.7992 - val_loss: 0.5905
Epoch 6/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 623s 796ms/step - accuracy: 0.7097 - loss: 0.8558 - val_accuracy: 0.8126 - val_loss: 0.5537
Epoch 7/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 621s 794ms/step - accuracy: 0.7283 - loss: 0.8028 - val_accuracy: 0.8193 - val_loss: 0.5255
Epoch 8/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 624s 797ms/step - accuracy: 0.7431 - loss: 0.7601 - val_accuracy: 0.8278 - val_loss: 0.5052
Epoch 9/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 617s 789ms/step - accuracy: 0.7546 - loss: 0.7223 - val_accuracy: 0.8314 - val_loss: 0.4862
Epoch 10/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 619s 791ms/step - accuracy: 0.7617 - loss: 0.6901 - val_accuracy: 0.8392 - val_loss: 0.4666
Epoch 11/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 622s 795ms/step - accuracy: 0.7733 - loss: 0.6743 - val_accuracy: 0.8429 - val_loss: 0.4556
Epoch 12/60
782/782 ━━━━━━━━━━━━━━━━━━━━ 618s 790ms/step - accuracy: 0.7789 - loss: 0.6448 - val_accuracy: 0.8499 - val_loss: 0.4408
Epoch 13/60
293/782 ━━━━━━━━━━━━━━━━━━━━ 6:22 782ms/step - accuracy: 0.7796 - loss: 0.6383
```

… still running

Learnings

- **modal depth & normalization:** accuracy improvement & stabilization
- **global average pooling:** reduces parameter
- **Adam** → faster; **SGD** → more stable
- **transfer learning:** drastically improves performance
- [**data augmentation:** essential to reduce overfitting]
- **validation gaps:** reveal need for better regularization or more diverse dat





Thank you!