dnn-retina-pro-1

April 24, 2025

[10]:
```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
 ↪Dropout, Input, LSTM, Reshape, GlobalAveragePooling2D
from tensorflow.keras.applications import VGG16, ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import time
```

[11]:
```
IMG_SIZE = 224  # Adjust this based on your dataset
BATCH_SIZE = 32
INPUT_SHAPE = (IMG_SIZE, IMG_SIZE, 3)
EPOCHS = 3  # Adjust number of epochs based on your dataset
```

[2]:
```
IMG_SIZE = 128
BATCH_SIZE = 32
INPUT_SHAPE = (IMG_SIZE, IMG_SIZE, 3)
EPOCHS = 3
```

[14]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Set image size and batch size
IMG_SIZE = (224, 224)  # You can change this to fit your model
BATCH_SIZE = 32



# Define paths to train and test directories (ensure the paths are correct)
# Update the paths to the absolute location
train_path = 'C:/Users/hp/Desktop/MTECH_ SEM 2/DNN/Diagnosis of Diabetic␣
 ↪Retinopathy/train'
test_path = 'C:/Users/hp/Desktop/MTECH_ SEM 2/DNN/Diagnosis of Diabetic␣
 ↪Retinopathy/test'
```

```
# Check if the directories exist (optional, for debugging)
if not os.path.exists(train_path):
    print(f"Train path does not exist: {train_path}")
else:
    print(f"Train path exists: {train_path}")

if not os.path.exists(test_path):
    print(f"Test path does not exist: {test_path}")
else:
    print(f"Test path exists: {test_path}")

# Initialize ImageDataGenerator
datagen = ImageDataGenerator(rescale=1./255)

# Load training data
train_data = datagen.flow_from_directory(
    train_path,
    target_size=IMG_SIZE,       # Resize images to IMG_SIZE
    batch_size=BATCH_SIZE,      # Define batch size
    class_mode='categorical'    # Multi-class classification
)

# Load testing data
test_data = datagen.flow_from_directory(
    test_path,
    target_size=IMG_SIZE,       # Resize images to IMG_SIZE
    batch_size=BATCH_SIZE,      # Define batch size
    class_mode='categorical',   # Multi-class classification
    shuffle=False               # Make sure test data isn't shuffled
)

# Number of classes (in your case it should be 2: DR and NO_DR)
NUM_CLASSES = train_data.num_classes
print(f"Number of classes: {NUM_CLASSES}")

# Optional: View class labels
print("Class labels:", train_data.class_indices)
```

```
Train path exists: C:/Users/hp/Desktop/MTECH_ SEM 2/DNN/Diagnosis of Diabetic
Retinopathy/train
Test path exists: C:/Users/hp/Desktop/MTECH_ SEM 2/DNN/Diagnosis of Diabetic
Retinopathy/test
Found 2076 images belonging to 2 classes.
Found 231 images belonging to 2 classes.
Number of classes: 2
```

```
Class labels: {'DR': 0, 'No_DR': 1}
```

[15]:
```python
# ------------------- MODEL BUILDING -------------------
def build_cnn():
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=INPUT_SHAPE),
        MaxPooling2D(2, 2),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D(2, 2),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(NUM_CLASSES, activation='softmax')
    ])
    return model

def build_dnn():
    model = Sequential([
        Flatten(input_shape=INPUT_SHAPE),
        Dense(256, activation='relu'),
        Dropout(0.5),
        Dense(NUM_CLASSES, activation='softmax')
    ])
    return model

def build_rnn():
    model = Sequential([
        Reshape((IMG_SIZE, IMG_SIZE * 3), input_shape=INPUT_SHAPE),
        LSTM(64, return_sequences=True),
        LSTM(64),
        Dense(NUM_CLASSES, activation='softmax')
    ])
    return model

def build_lenet():
    model = Sequential([
        Conv2D(6, (5, 5), activation='relu', input_shape=INPUT_SHAPE),
        MaxPooling2D(),
        Conv2D(16, (5, 5), activation='relu'),
        MaxPooling2D(),
        Flatten(),
        Dense(120, activation='relu'),
        Dense(84, activation='relu'),
        Dense(NUM_CLASSES, activation='softmax')
    ])
    return model
```

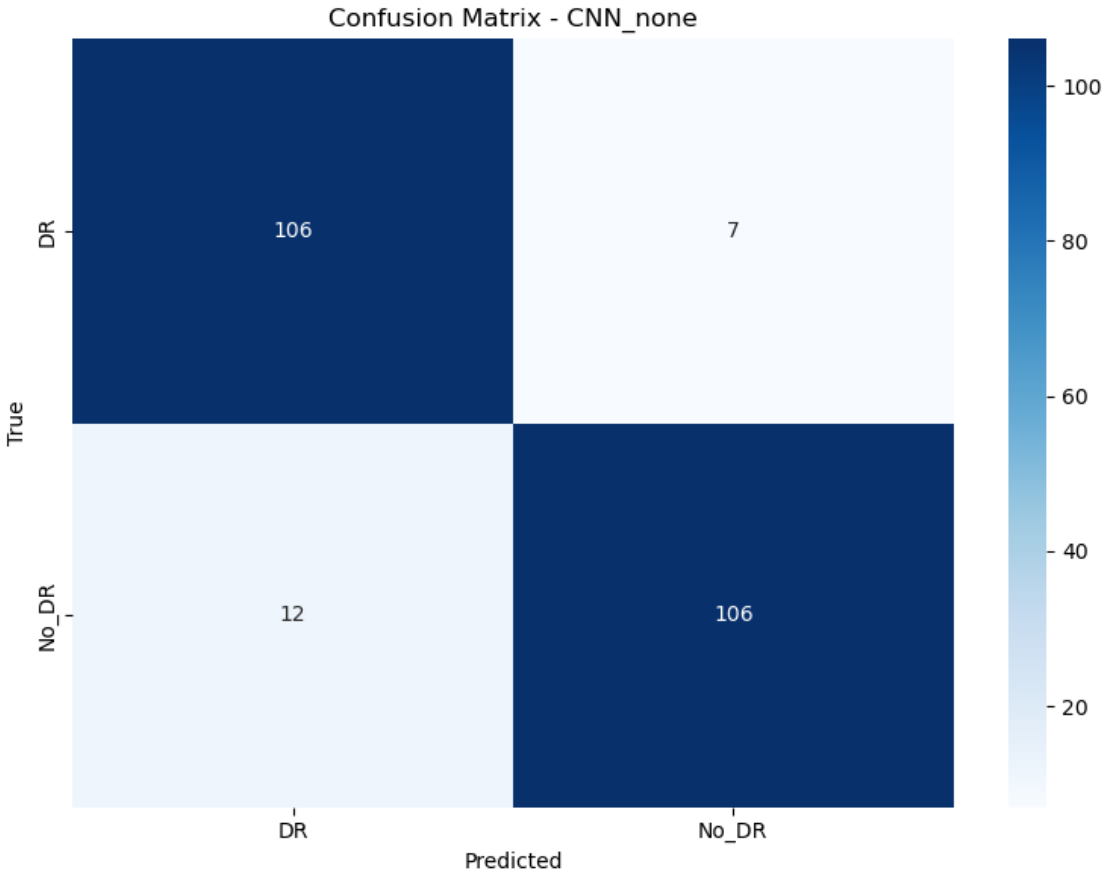
```
def build_vgg16():
    base = VGG16(weights='imagenet', include_top=False, input_shape=INPUT_SHAPE)
    for layer in base.layers:
        layer.trainable = False
    x = base.output
    x = Flatten()(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.5)(x)
    preds = Dense(NUM_CLASSES, activation='softmax')(x)
    model = Model(inputs=base.input, outputs=preds)
    return model

def build_resnet():
    base = ResNet50(weights='imagenet', include_top=False,␣
 ↪input_shape=INPUT_SHAPE)
    for layer in base.layers:
        layer.trainable = False
    x = base.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(128, activation='relu')(x)
    preds = Dense(NUM_CLASSES, activation='softmax')(x)
    model = Model(inputs=base.input, outputs=preds)
    return model
```

[17]:
```
# ------------------- TRAINING FRAMEWORK -------------------
models = {
    "CNN": build_cnn,
    "DNN": build_dnn,
    "RNN": build_rnn,
    "LeNet": build_lenet,
    "VGG16": build_vgg16,
    "ResNet": build_resnet
}

optimizers = {
    "adam": Adam,
    "sgd": SGD,
    "rmsprop": RMSprop
}

learning_rates = [0.001, 0.01]

results = []
history_logs = {}
model_summaries = {}
speed_logs = {}
```

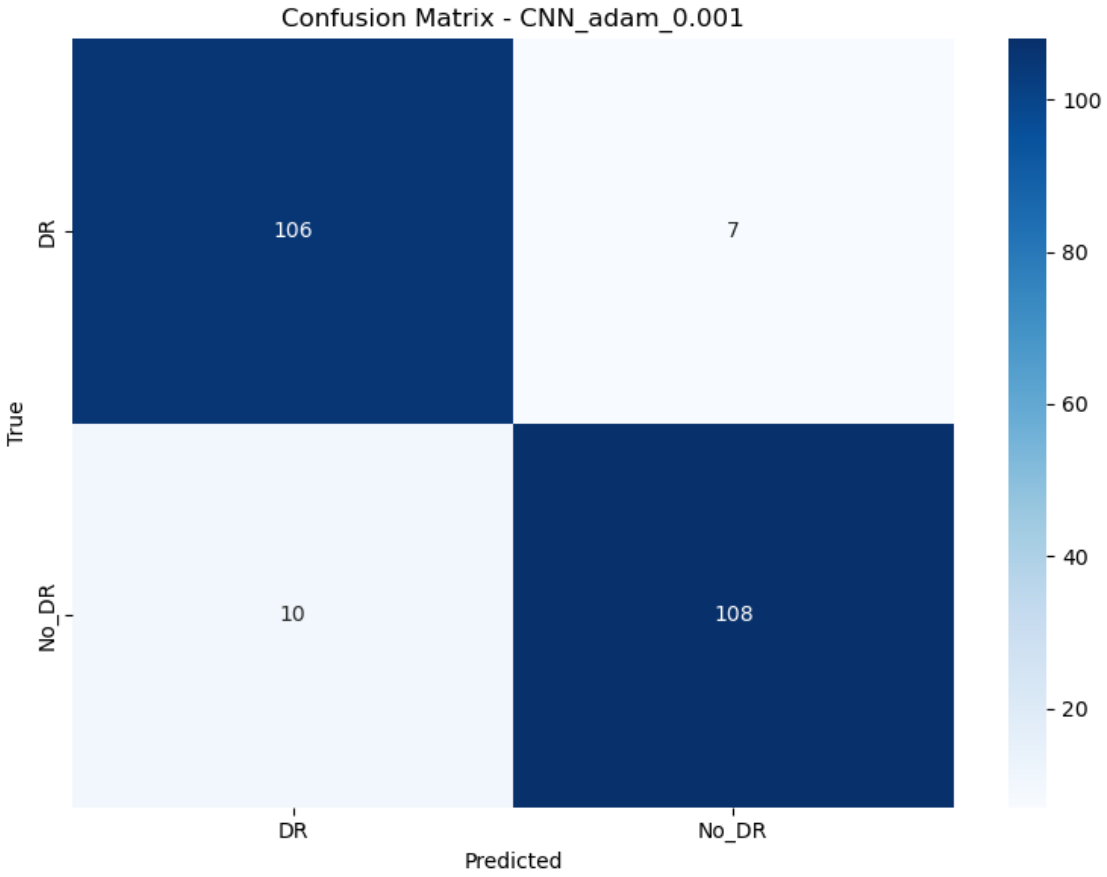
```
def plot_confusion_matrix(y_true, y_pred, classes, title):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes,␣
 ↪yticklabels=classes)
    plt.title(f'Confusion Matrix - {title}')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.tight_layout()
    plt.show()

def train_and_evaluate(model_fn, optimizer_fn, lr, model_name, label):
    model = model_fn()
    model_summaries[f"{model_name}_{label}"] = [layer.__class__.__name__ for␣
 ↪layer in model.layers]

    if optimizer_fn:
        optimizer = optimizer_fn(learning_rate=lr)
        model.compile(optimizer=optimizer, loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
    else:
        model.compile(loss='categorical_crossentropy', metrics=['accuracy'])

    start_time = time.time()
    history = model.fit(train_data, epochs=EPOCHS, validation_data=test_data,␣
 ↪verbose=0)
    duration = time.time() - start_time

    loss, acc = model.evaluate(test_data, verbose=0)
    history_logs[f"{model_name}_{label}"] = history
    speed_logs[f"{model_name}_{label}"] = duration

    preds = model.predict(test_data)
    y_pred = np.argmax(preds, axis=1)
    y_true = test_data.classes
    class_labels = list(test_data.class_indices.keys())

    # Confusion Matrix
    plot_confusion_matrix(y_true, y_pred, class_labels, f"{model_name}_{label}")

    # Classification Report
    print(f"Classification Report - {model_name}_{label}:")
    print(classification_report(y_true, y_pred, target_names=class_labels))

    return loss, acc, model

trained_models = {}
```

```
for model_name, model_fn in models.items():
    print(f"===== Training {model_name} WITHOUT Optimizer =====")
    try:
        loss, acc, model = train_and_evaluate(model_fn, None, None, model_name,␣
 ↪"none")
        results.append((model_name, "None", "None", loss, acc))
        trained_models[f"{model_name}_none"] = model
        print(f"  {model_name} no optimizer - Loss: {loss:.4f}, Accuracy: {acc:.
 ↪4f}")
    except Exception as e:
        print(f"  Error in {model_name} without optimizer: {e}")

    for opt_name, opt_class in optimizers.items():
        for lr in learning_rates:
            print(f"===== Training {model_name} with {opt_name.upper()}␣
 ↪(lr={lr}) =====")
            try:
                loss, acc, model = train_and_evaluate(model_fn, opt_class, lr,␣
 ↪model_name, f"{opt_name}_{lr}")
                results.append((model_name, opt_name, lr, loss, acc))
                trained_models[f"{model_name}_{opt_name}_{lr}"] = model
                print(f"  {model_name} + {opt_name} - Loss: {loss:.4f},␣
 ↪Accuracy: {acc:.4f}")
            except Exception as e:
                print(f"  Error in {model_name} with {opt_name} (lr={lr}): {e}")
```

```
===== Training CNN WITHOUT Optimizer =====
8/8             2s 172ms/step
```

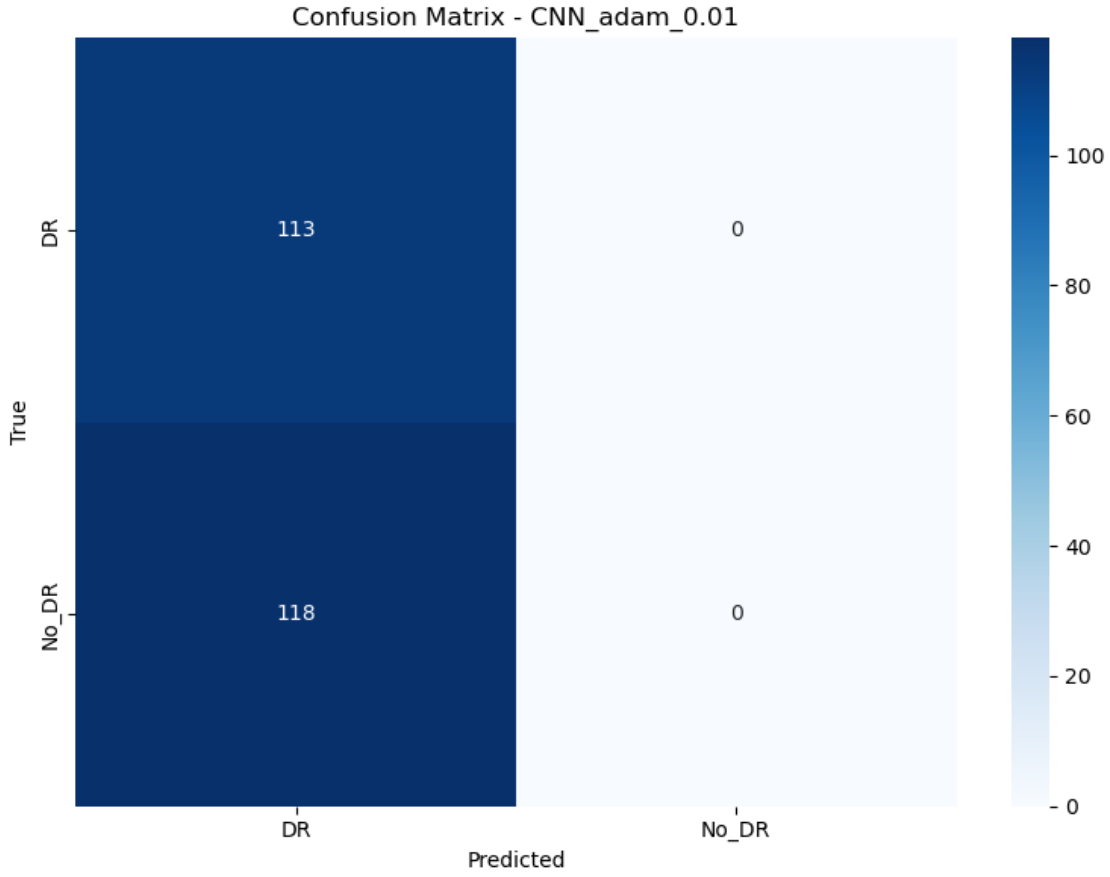



```
Classification Report - CNN_none:
              precision    recall  f1-score   support

          DR       0.90      0.94      0.92       113
       No_DR       0.94      0.90      0.92       118

    accuracy                           0.92       231
   macro avg       0.92      0.92      0.92       231
weighted avg       0.92      0.92      0.92       231

  CNN no optimizer - Loss: 0.2333, Accuracy: 0.9177
===== Training CNN with ADAM (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8            2s 178ms/step
```



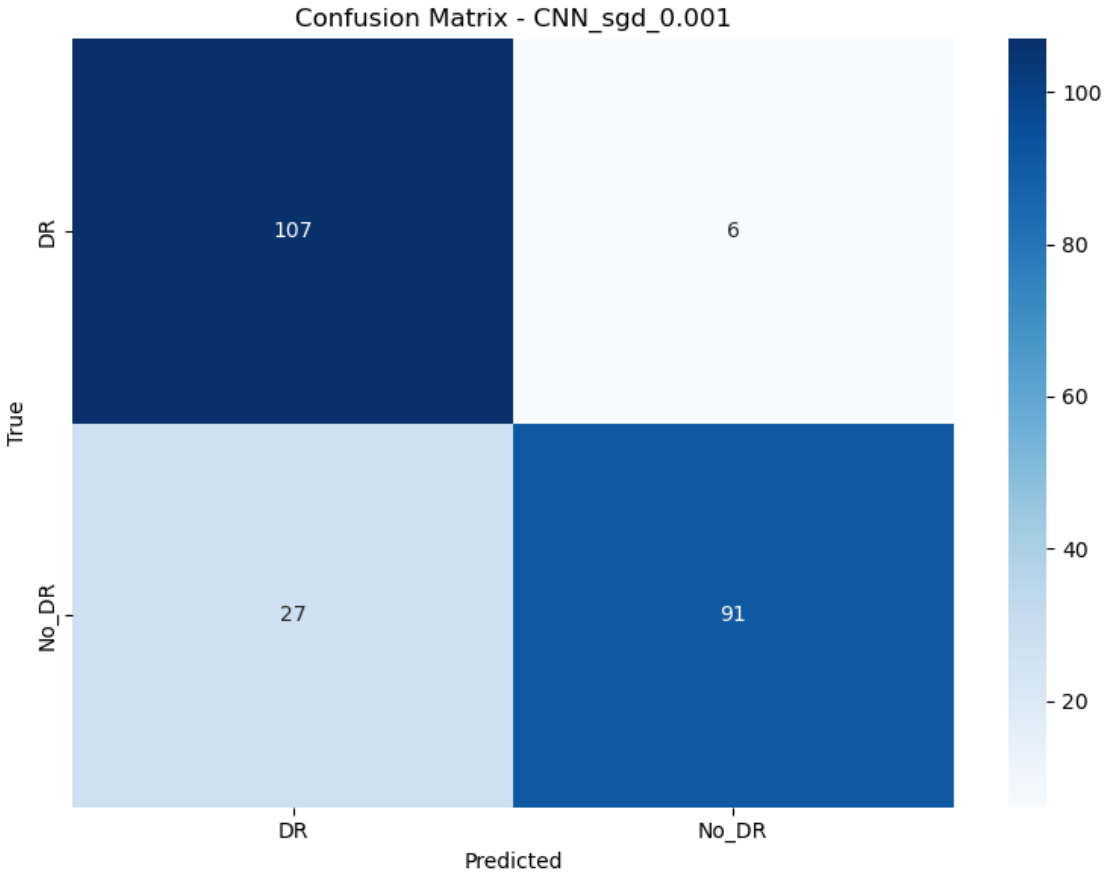

```
Classification Report - CNN_adam_0.001:
              precision    recall  f1-score   support

          DR       0.91      0.94      0.93       113
       No_DR       0.94      0.92      0.93       118

    accuracy                           0.93       231
   macro avg       0.93      0.93      0.93       231
weighted avg       0.93      0.93      0.93       231

  CNN + adam - Loss: 0.2087, Accuracy: 0.9264
===== Training CNN with ADAM (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

WARNING:tensorflow:5 out of the last 17 calls to <function
```

```
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x0000017A1A8515A0> triggered tf.function retracing. Tracing is expensive and
the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
8/8             2s 181ms/step
```

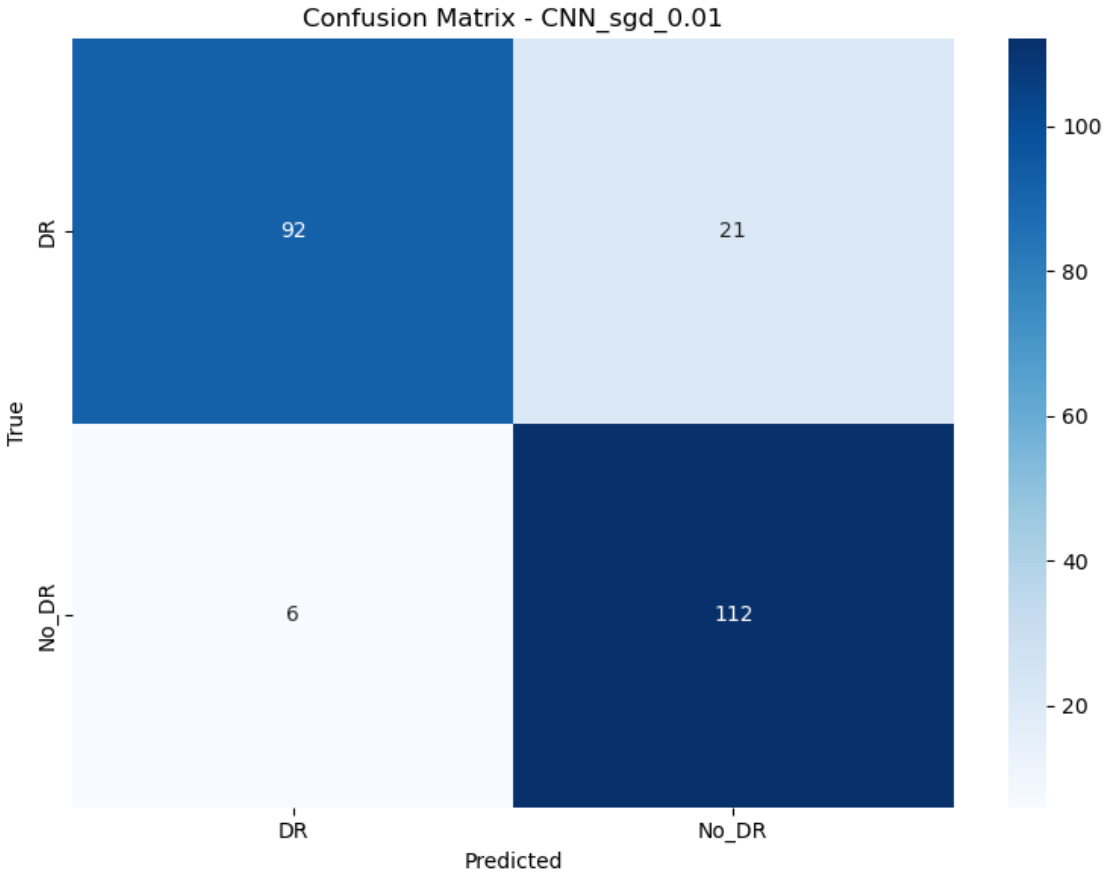



```
Classification Report - CNN_adam_0.01:
              precision    recall  f1-score   support

          DR       0.49      1.00      0.66       113
       No_DR       0.00      0.00      0.00       118

    accuracy                           0.49       231
   macro avg       0.24      0.50      0.33       231
weighted avg       0.24      0.49      0.32       231
```

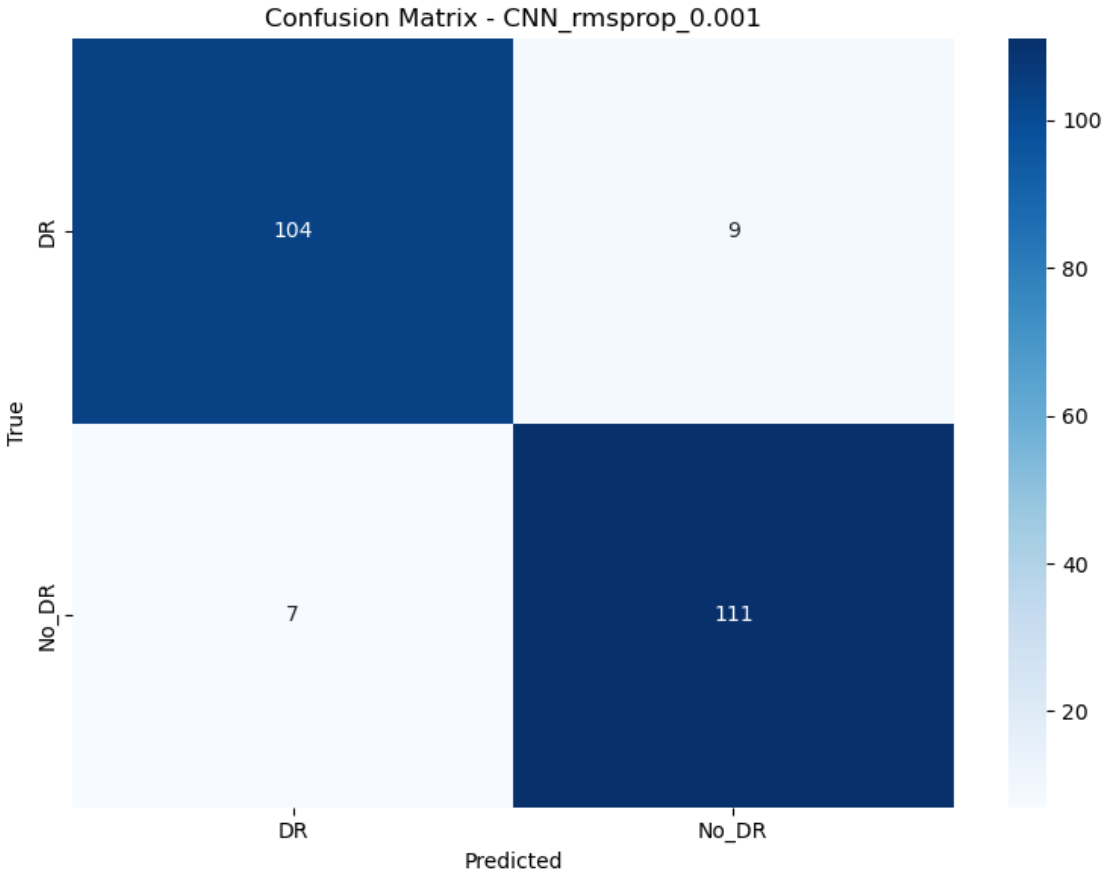
```
  CNN + adam - Loss: 0.6944, Accuracy: 0.4892
===== Training CNN with SGD (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

WARNING:tensorflow:5 out of the last 17 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x0000017A1A5F65F0> triggered tf.function retracing. Tracing is expensive and
the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
8/8            2s 175ms/step
```



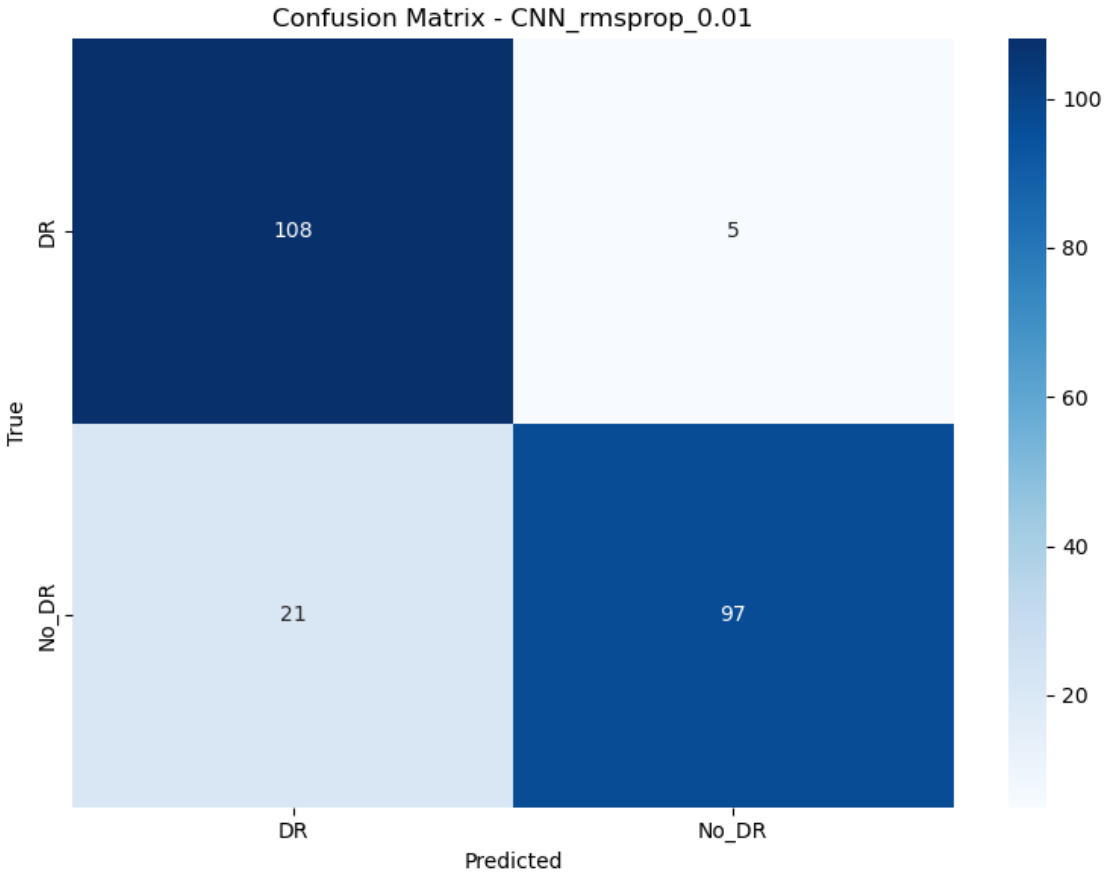

```
Classification Report - CNN_sgd_0.001:
              precision    recall  f1-score   support

          DR       0.80      0.95      0.87       113
       No_DR       0.94      0.77      0.85       118

    accuracy                           0.86       231
   macro avg       0.87      0.86      0.86       231
weighted avg       0.87      0.86      0.86       231

  CNN + sgd - Loss: 0.3591, Accuracy: 0.8571
===== Training CNN with SGD (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8             2s 184ms/step
```



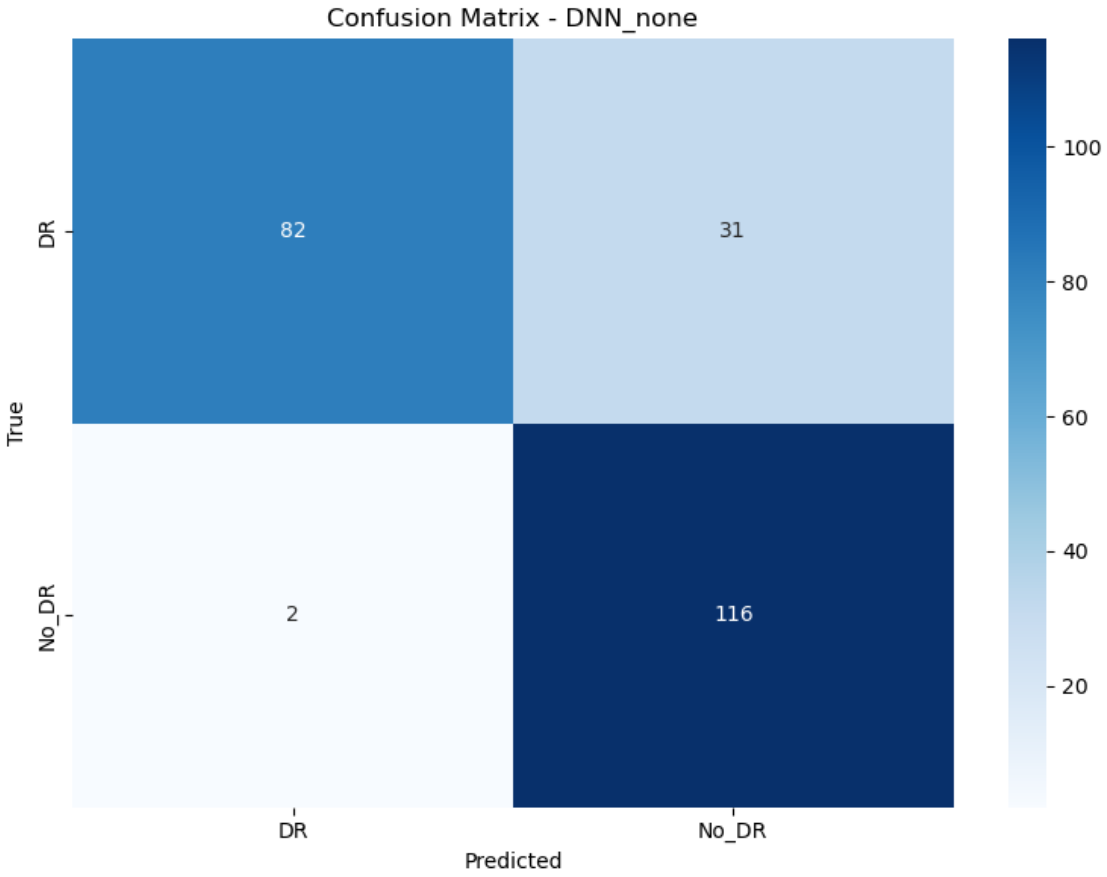

```
Classification Report - CNN_sgd_0.01:
              precision    recall  f1-score   support

          DR       0.94      0.81      0.87       113
       No_DR       0.84      0.95      0.89       118

    accuracy                           0.88       231
   macro avg       0.89      0.88      0.88       231
weighted avg       0.89      0.88      0.88       231

  CNN + sgd - Loss: 0.3463, Accuracy: 0.8831
===== Training CNN with RMSPROP (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8            2s 229ms/step
```



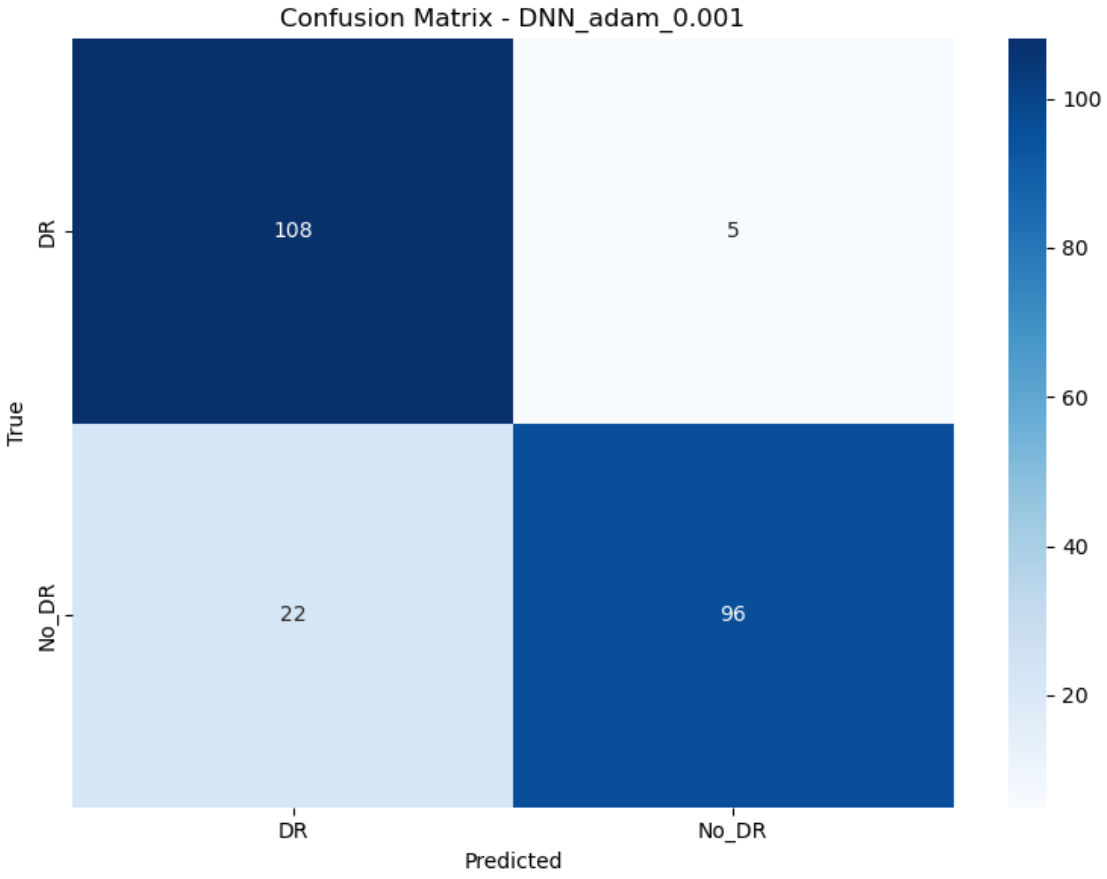

```
Classification Report - CNN_rmsprop_0.001:
              precision    recall  f1-score   support

          DR       0.94      0.92      0.93       113
       No_DR       0.93      0.94      0.93       118

    accuracy                           0.93       231
   macro avg       0.93      0.93      0.93       231
weighted avg       0.93      0.93      0.93       231

  CNN + rmsprop - Loss: 0.2043, Accuracy: 0.9307
===== Training CNN with RMSPROP (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8            2s 176ms/step
```



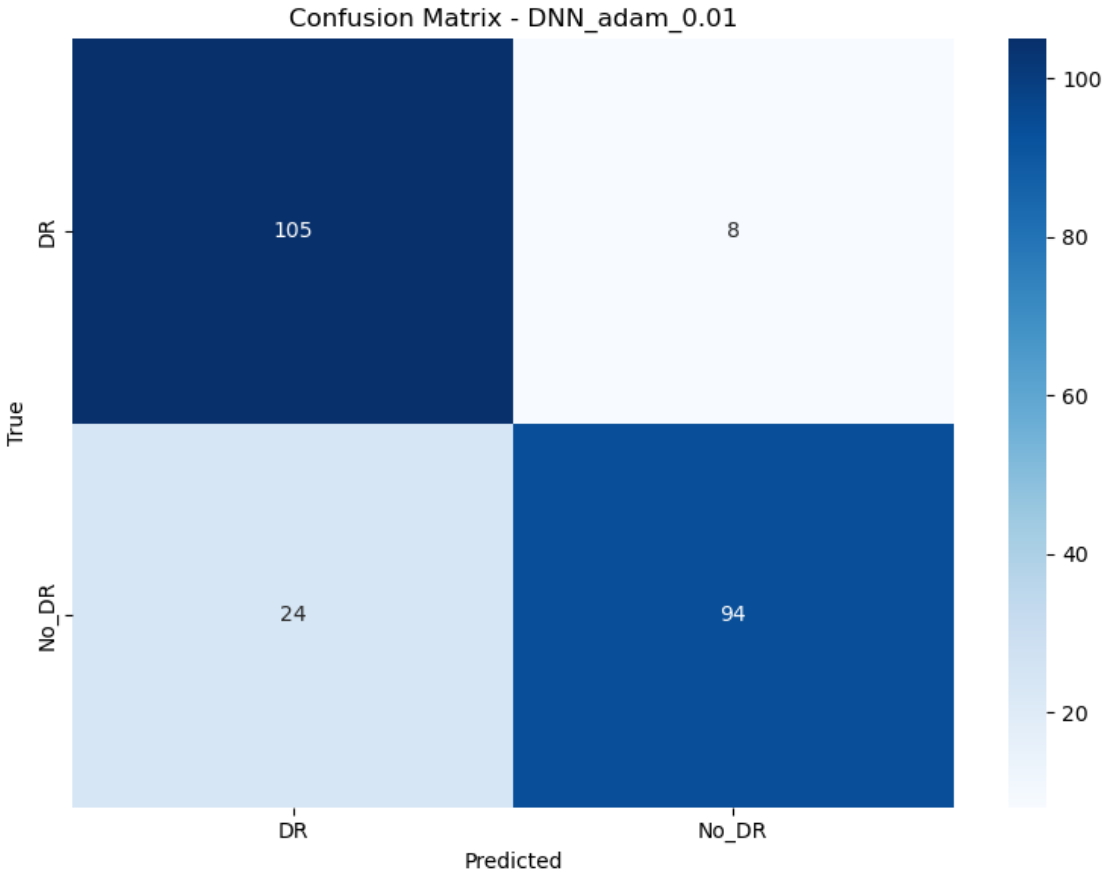

```
Classification Report - CNN_rmsprop_0.01:
              precision    recall  f1-score   support

          DR       0.84      0.96      0.89       113
       No_DR       0.95      0.82      0.88       118

    accuracy                           0.89       231
   macro avg       0.89      0.89      0.89       231
weighted avg       0.90      0.89      0.89       231

  CNN + rmsprop - Loss: 0.2930, Accuracy: 0.8874
===== Training DNN WITHOUT Optimizer =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

8/8            1s 122ms/step
```



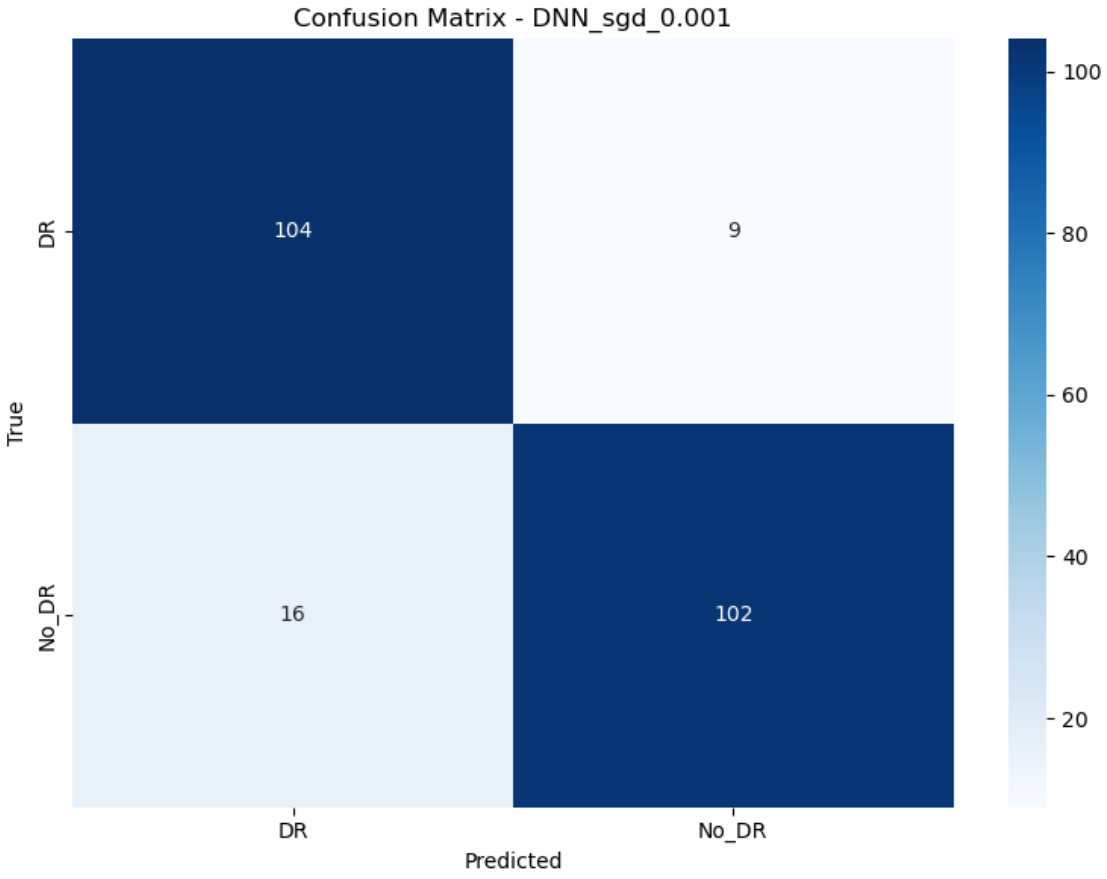

```
Classification Report - DNN_none:
              precision    recall  f1-score   support

          DR       0.98      0.73      0.83       113
       No_DR       0.79      0.98      0.88       118

    accuracy                           0.86       231
   macro avg       0.88      0.85      0.85       231
weighted avg       0.88      0.86      0.85       231

  DNN no optimizer - Loss: 0.3494, Accuracy: 0.8571
===== Training DNN with ADAM (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

8/8            1s 124ms/step
```



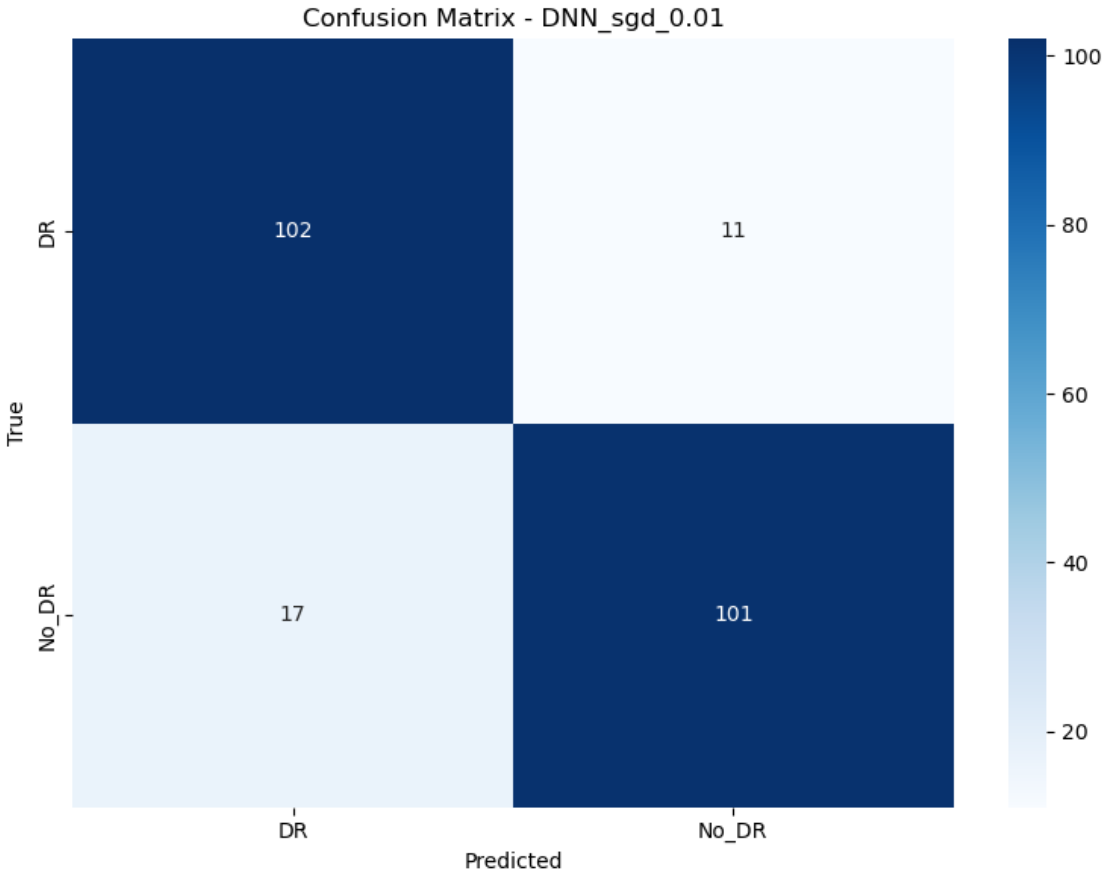

```
Classification Report - DNN_adam_0.001:
              precision    recall  f1-score   support

          DR       0.83      0.96      0.89       113
       No_DR       0.95      0.81      0.88       118

    accuracy                           0.88       231
   macro avg       0.89      0.88      0.88       231
weighted avg       0.89      0.88      0.88       231

  DNN + adam - Loss: 0.2519, Accuracy: 0.8831
===== Training DNN with ADAM (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

8/8            1s 114ms/step
```



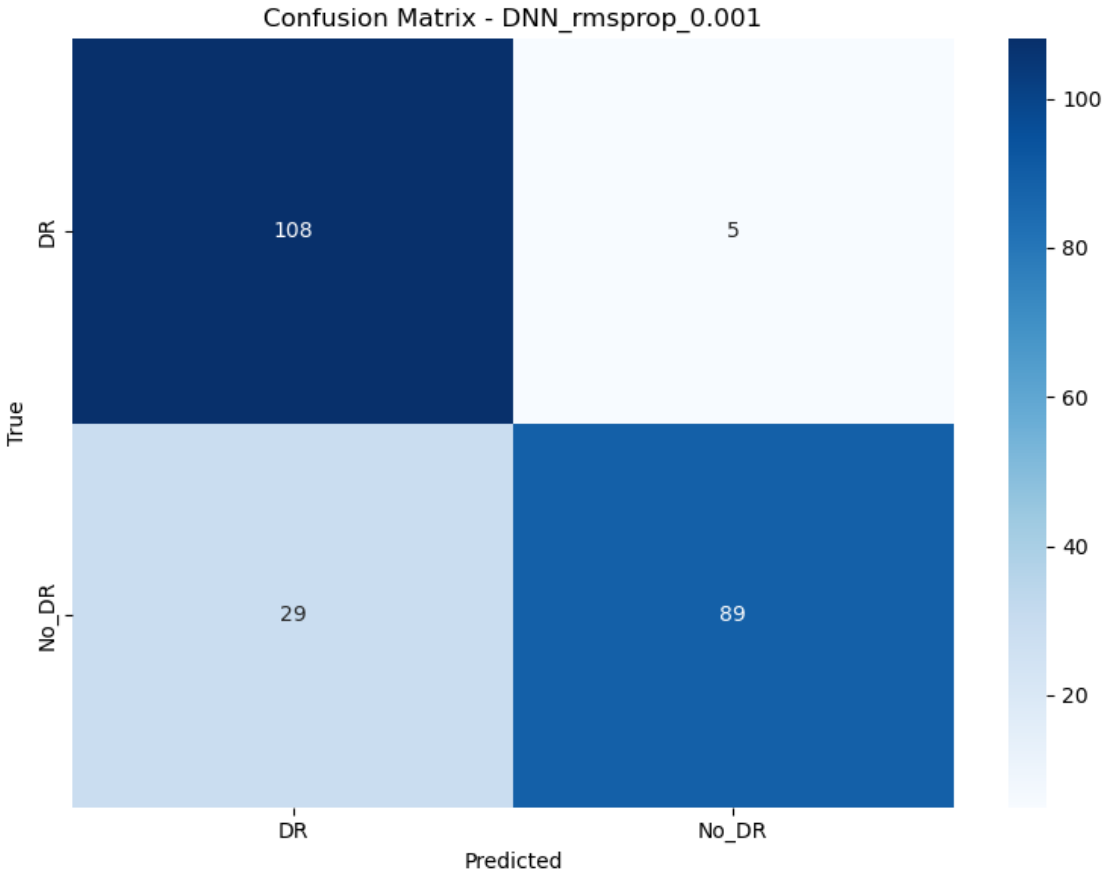

```
Classification Report - DNN_adam_0.01:
              precision    recall  f1-score   support

          DR       0.81      0.93      0.87       113
       No_DR       0.92      0.80      0.85       118

    accuracy                           0.86       231
   macro avg       0.87      0.86      0.86       231
weighted avg       0.87      0.86      0.86       231

  DNN + adam - Loss: 0.3228, Accuracy: 0.8615
===== Training DNN with SGD (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

8/8            1s 111ms/step
```



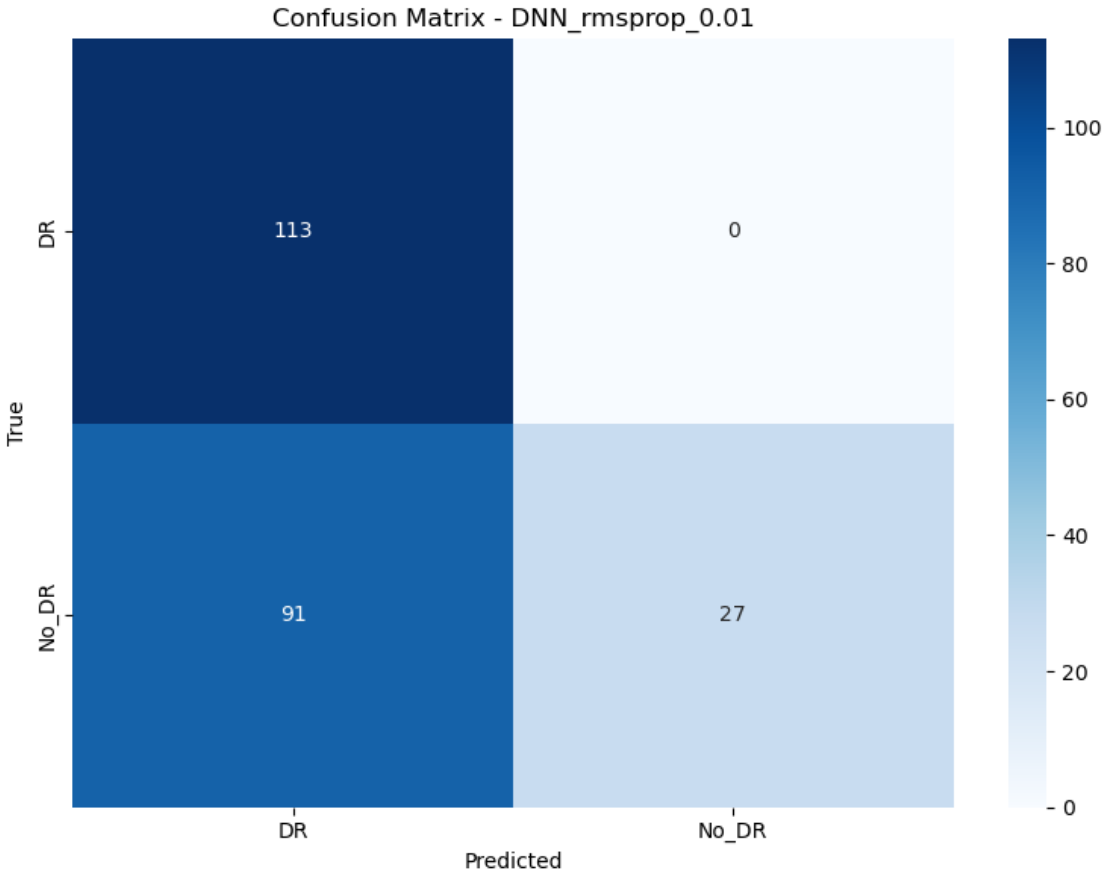

```
Classification Report - DNN_sgd_0.001:
              precision    recall  f1-score   support

          DR       0.87      0.92      0.89       113
       No_DR       0.92      0.86      0.89       118

    accuracy                           0.89       231
   macro avg       0.89      0.89      0.89       231
weighted avg       0.89      0.89      0.89       231

  DNN + sgd - Loss: 0.2617, Accuracy: 0.8918
===== Training DNN with SGD (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

8/8            1s 110ms/step
```




```
Classification Report - DNN_sgd_0.01:
              precision    recall  f1-score   support

          DR       0.86      0.90      0.88       113
       No_DR       0.90      0.86      0.88       118

    accuracy                           0.88       231
   macro avg       0.88      0.88      0.88       231
weighted avg       0.88      0.88      0.88       231

  DNN + sgd - Loss: 0.2640, Accuracy: 0.8788
===== Training DNN with RMSPROP (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

8/8            1s 118ms/step
```

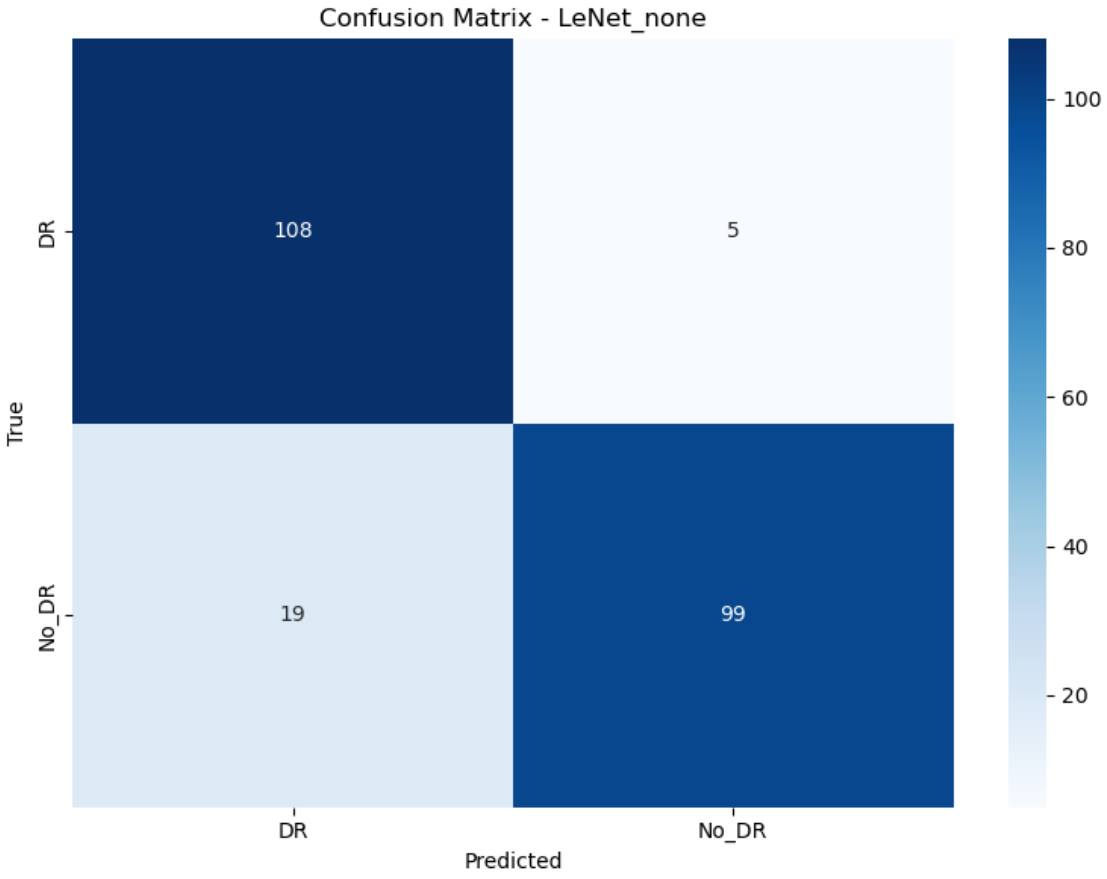



```
Classification Report - DNN_rmsprop_0.001:
              precision    recall  f1-score   support

          DR       0.79      0.96      0.86       113
       No_DR       0.95      0.75      0.84       118

    accuracy                           0.85       231
   macro avg       0.87      0.85      0.85       231
weighted avg       0.87      0.85      0.85       231

  DNN + rmsprop - Loss: 0.2861, Accuracy: 0.8528
===== Training DNN with RMSPROP (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

8/8            1s 122ms/step
```

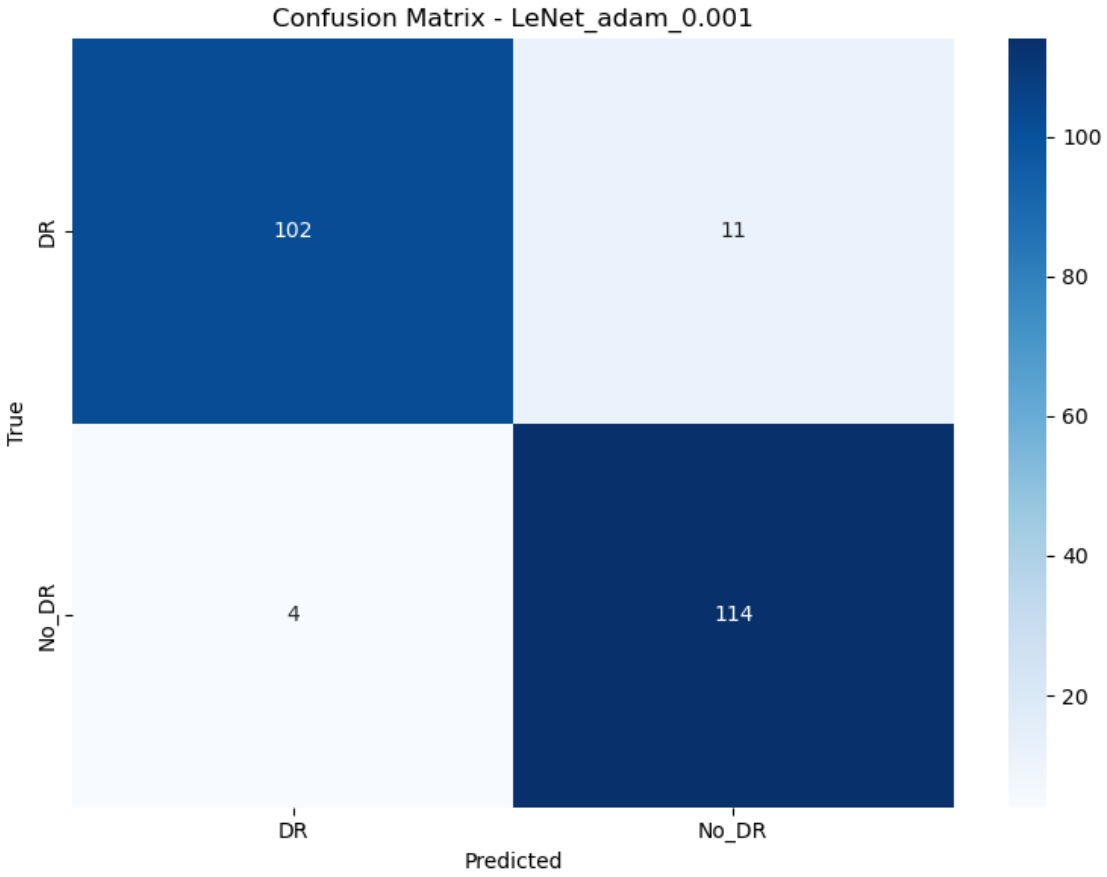



```
Classification Report - DNN_rmsprop_0.01:
              precision    recall  f1-score   support

          DR       0.55      1.00      0.71       113
       No_DR       1.00      0.23      0.37       118

    accuracy                           0.61       231
   macro avg       0.78      0.61      0.54       231
weighted avg       0.78      0.61      0.54       231

  DNN + rmsprop - Loss: 0.6075, Accuracy: 0.6061
===== Training RNN WITHOUT Optimizer =====
  Error in RNN without optimizer: can't multiply sequence by non-int of type
'tuple'
===== Training RNN with ADAM (lr=0.001) =====
  Error in RNN with adam (lr=0.001): can't multiply sequence by non-int of type
'tuple'
===== Training RNN with ADAM (lr=0.01) =====
  Error in RNN with adam (lr=0.01): can't multiply sequence by non-int of type
'tuple'
```

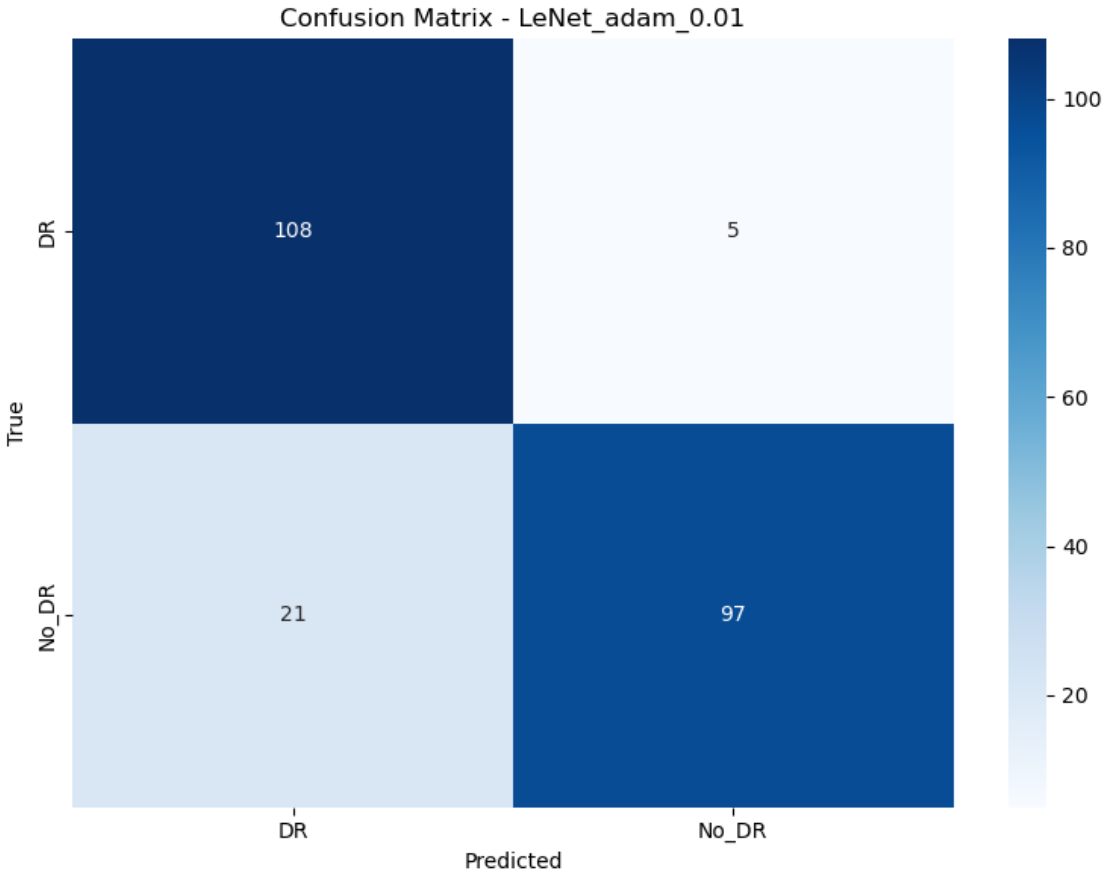
```
===== Training RNN with SGD (lr=0.001) =====
 Error in RNN with sgd (lr=0.001): can't multiply sequence by non-int of type
'tuple'
===== Training RNN with SGD (lr=0.01) =====
 Error in RNN with sgd (lr=0.01): can't multiply sequence by non-int of type
'tuple'
===== Training RNN with RMSPROP (lr=0.001) =====
 Error in RNN with rmsprop (lr=0.001): can't multiply sequence by non-int of
type 'tuple'
===== Training RNN with RMSPROP (lr=0.01) =====
 Error in RNN with rmsprop (lr=0.01): can't multiply sequence by non-int of
type 'tuple'
===== Training LeNet WITHOUT Optimizer =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\reshaping\reshape.py:39: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8             1s 127ms/step
```



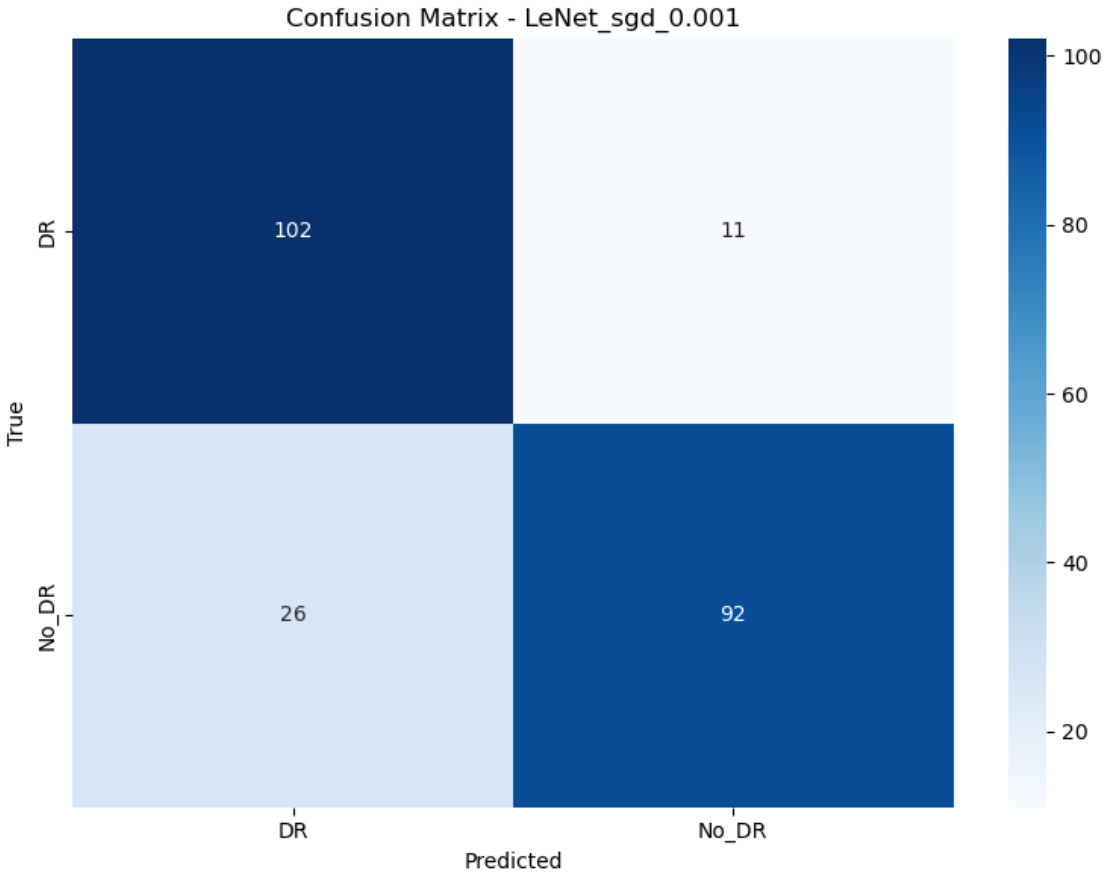

```
Classification Report - LeNet_none:
              precision    recall  f1-score   support

          DR       0.85      0.96      0.90       113
       No_DR       0.95      0.84      0.89       118

    accuracy                           0.90       231
   macro avg       0.90      0.90      0.90       231
weighted avg       0.90      0.90      0.90       231

  LeNet no optimizer - Loss: 0.2444, Accuracy: 0.8961
===== Training LeNet with ADAM (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8            1s 123ms/step
```



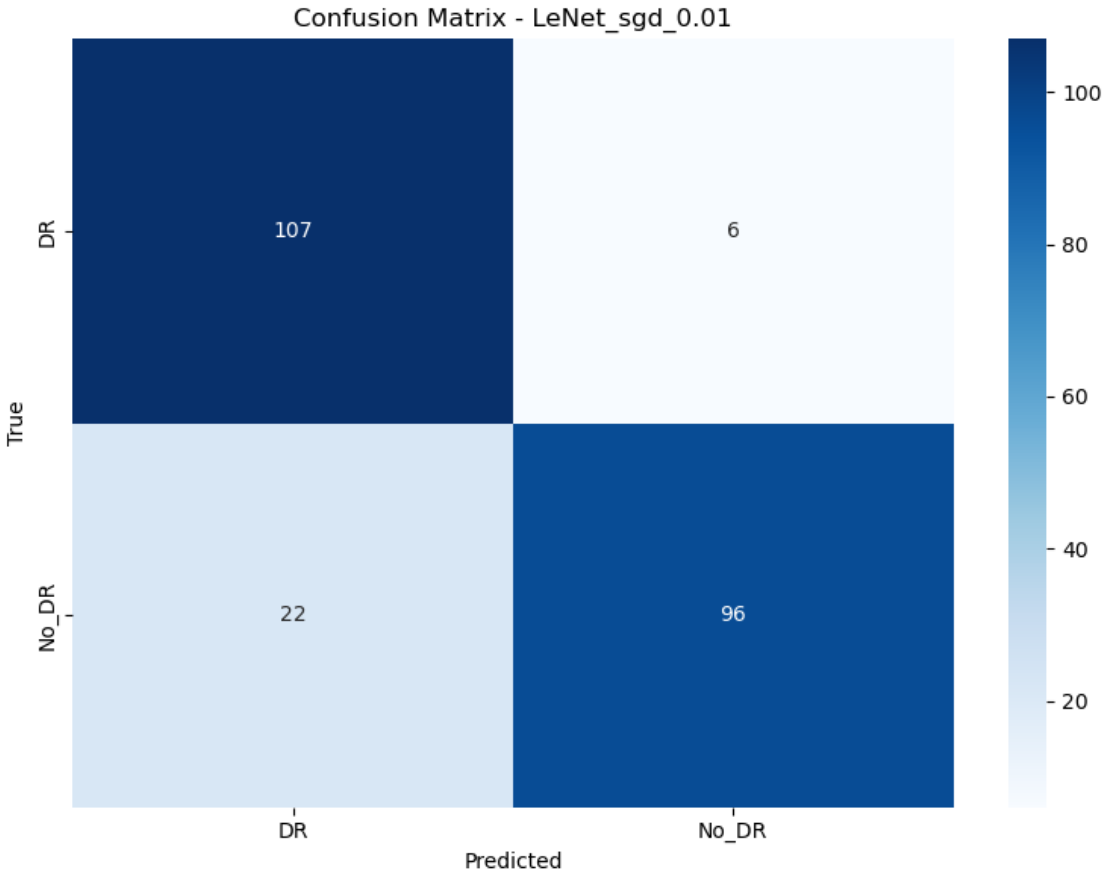

```
Classification Report - LeNet_adam_0.001:
              precision    recall  f1-score   support

          DR       0.96      0.90      0.93       113
       No_DR       0.91      0.97      0.94       118

    accuracy                           0.94       231
   macro avg       0.94      0.93      0.93       231
weighted avg       0.94      0.94      0.93       231

  LeNet + adam - Loss: 0.1820, Accuracy: 0.9351
===== Training LeNet with ADAM (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8             1s 123ms/step
```



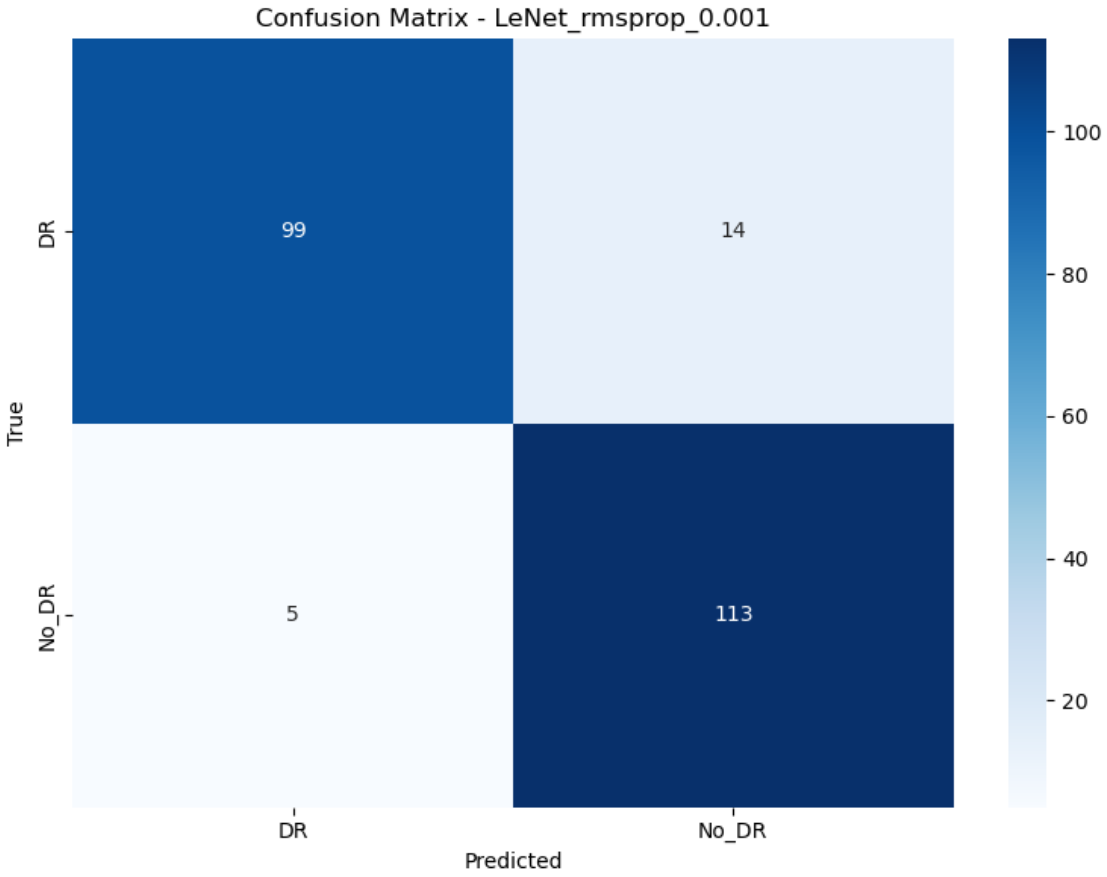

```
Classification Report - LeNet_adam_0.01:
              precision    recall  f1-score   support

          DR       0.84      0.96      0.89       113
       No_DR       0.95      0.82      0.88       118

    accuracy                           0.89       231
   macro avg       0.89      0.89      0.89       231
weighted avg       0.90      0.89      0.89       231

  LeNet + adam - Loss: 0.2197, Accuracy: 0.8874
===== Training LeNet with SGD (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8            1s 130ms/step
```



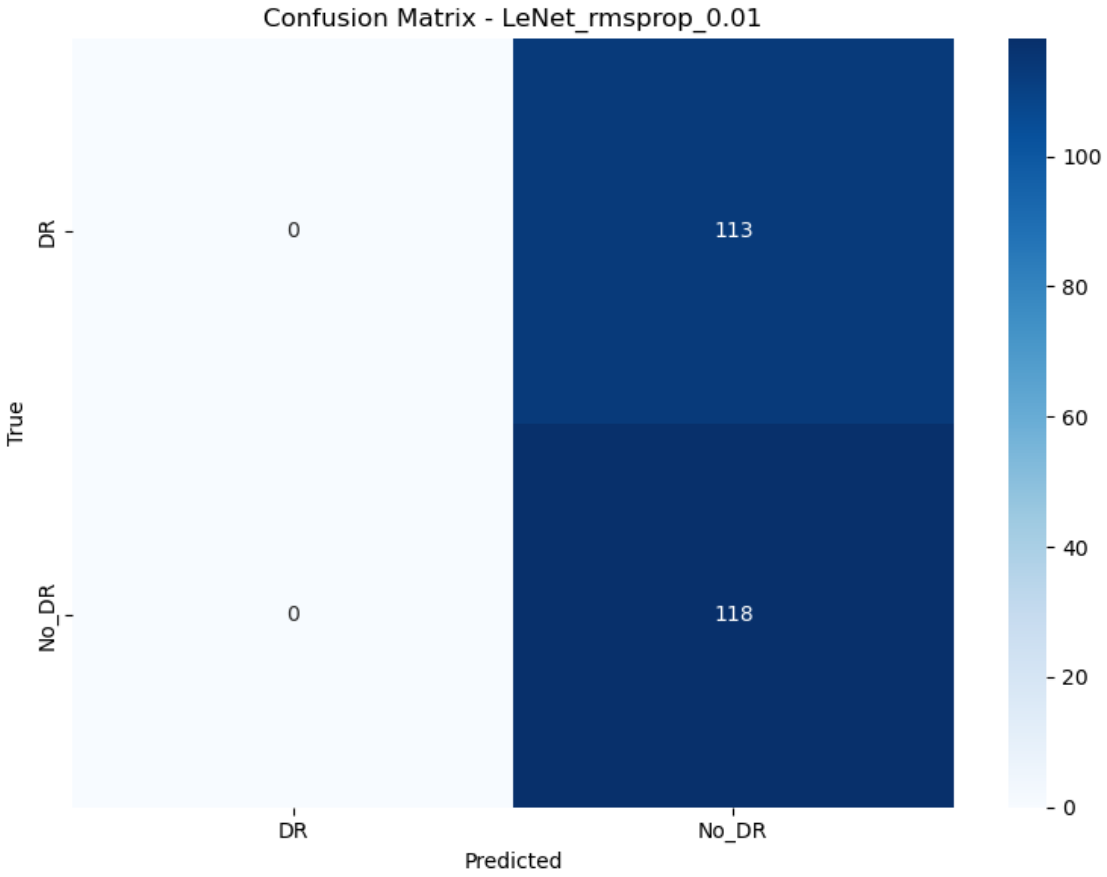

```
Classification Report - LeNet_sgd_0.001:
              precision    recall  f1-score   support

          DR       0.80      0.90      0.85       113
       No_DR       0.89      0.78      0.83       118

    accuracy                           0.84       231
   macro avg       0.85      0.84      0.84       231
weighted avg       0.85      0.84      0.84       231

  LeNet + sgd - Loss: 0.6111, Accuracy: 0.8398
===== Training LeNet with SGD (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8            1s 131ms/step
```




```
Classification Report - LeNet_sgd_0.01:
              precision    recall  f1-score   support

          DR       0.83      0.95      0.88       113
       No_DR       0.94      0.81      0.87       118

    accuracy                           0.88       231
   macro avg       0.89      0.88      0.88       231
weighted avg       0.89      0.88      0.88       231

  LeNet + sgd - Loss: 0.2571, Accuracy: 0.8788
===== Training LeNet with RMSPROP (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8            1s 130ms/step
```

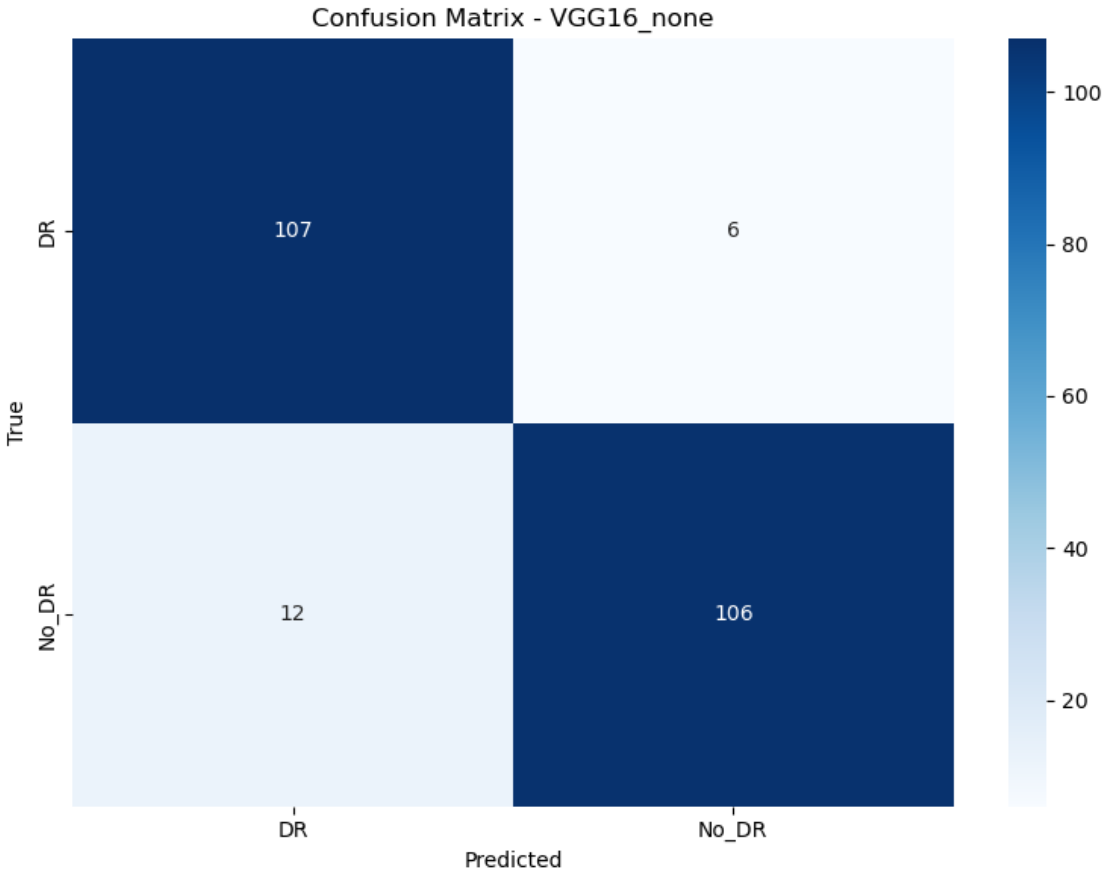



```
Classification Report - LeNet_rmsprop_0.001:
              precision    recall  f1-score   support

          DR       0.95      0.88      0.91       113
       No_DR       0.89      0.96      0.92       118

    accuracy                           0.92       231
   macro avg       0.92      0.92      0.92       231
weighted avg       0.92      0.92      0.92       231

  LeNet + rmsprop - Loss: 0.2142, Accuracy: 0.9177
===== Training LeNet with RMSPROP (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

8/8             1s 135ms/step
```



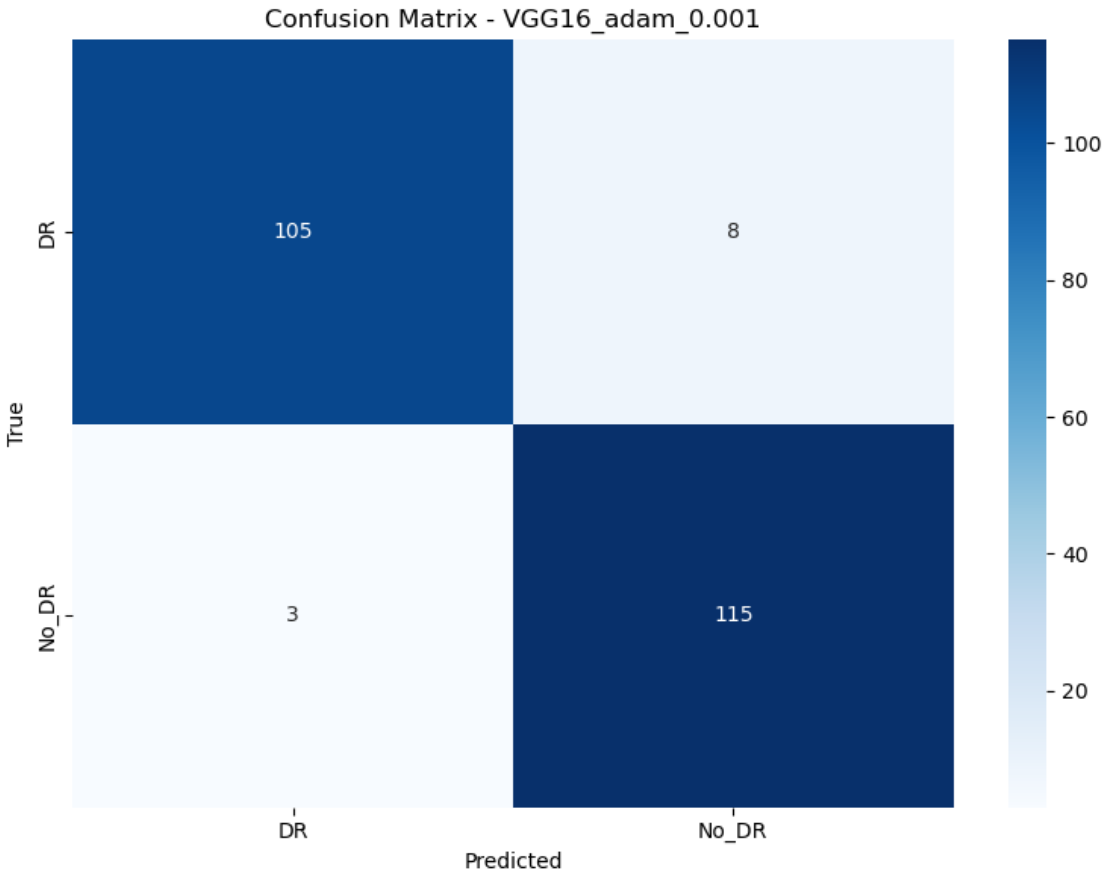

```
Classification Report - LeNet_rmsprop_0.01:
              precision    recall  f1-score   support

          DR       0.00      0.00      0.00       113
       No_DR       0.51      1.00      0.68       118

    accuracy                           0.51       231
   macro avg       0.26      0.50      0.34       231
weighted avg       0.26      0.51      0.35       231

  LeNet + rmsprop - Loss: 13.1654, Accuracy: 0.5108
===== Training VGG16 WITHOUT Optimizer =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
```

```
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256             52s
1us/step
8/8             48s 6s/step
```



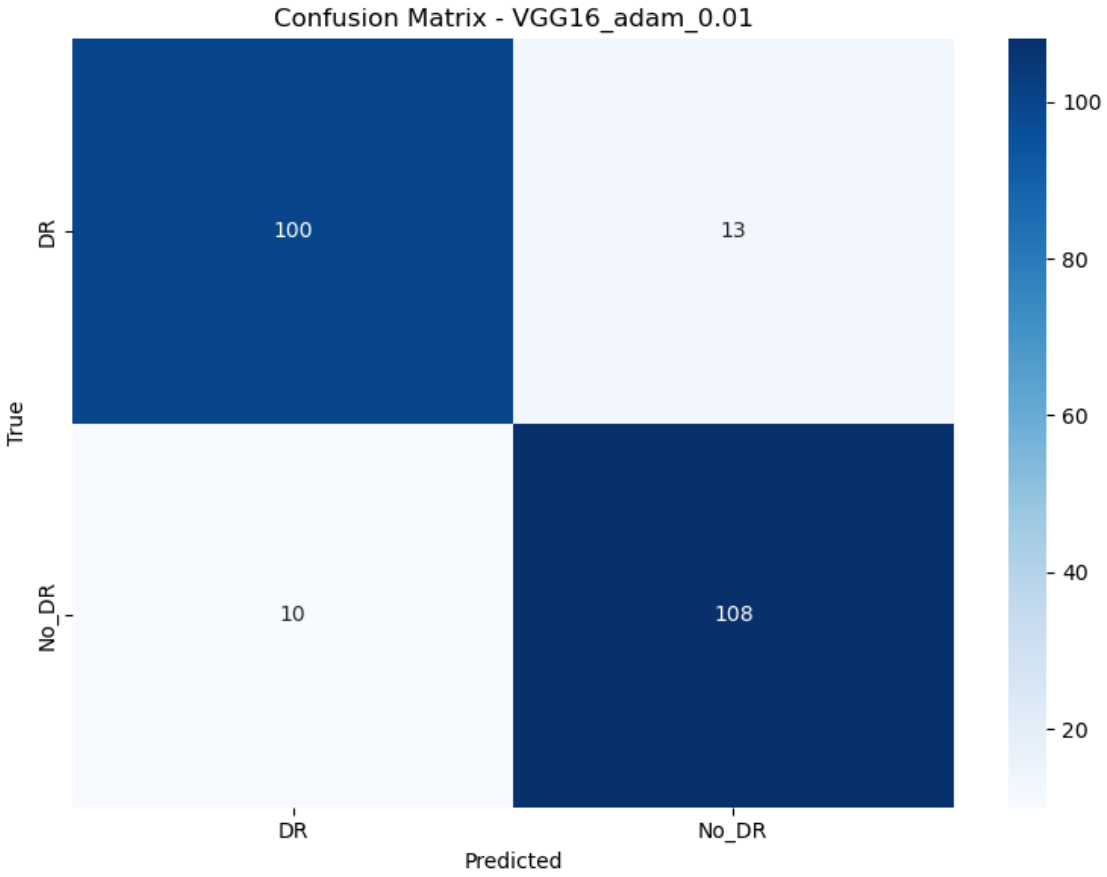

```
Classification Report - VGG16_none:
              precision    recall  f1-score   support

          DR       0.90      0.95      0.92       113
       No_DR       0.95      0.90      0.92       118
```

```
    accuracy                           0.92       231
   macro avg       0.92      0.92      0.92       231
weighted avg       0.92      0.92      0.92       231

  VGG16 no optimizer - Loss: 0.1975, Accuracy: 0.9221
===== Training VGG16 with ADAM (lr=0.001) =====
8/8             46s 6s/step
```

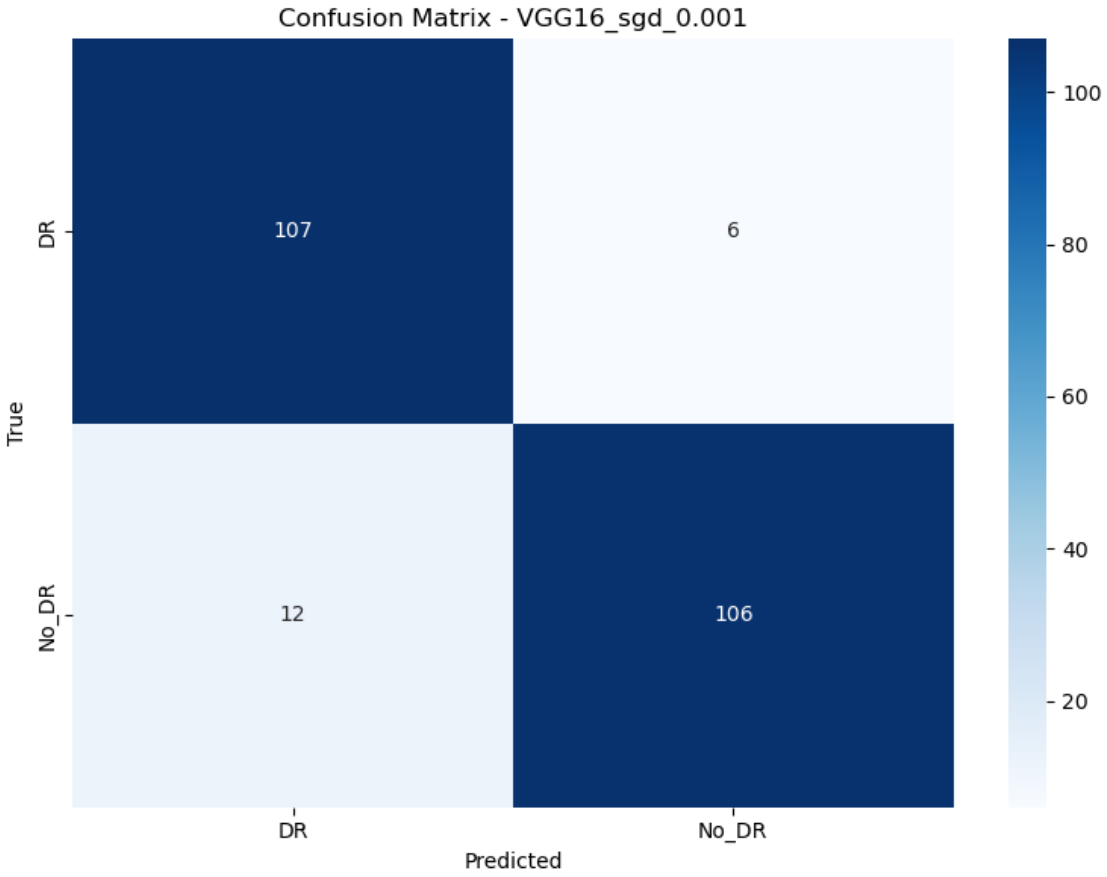



```
Classification Report - VGG16_adam_0.001:
              precision    recall  f1-score   support

          DR       0.97      0.93      0.95       113
       No_DR       0.93      0.97      0.95       118

    accuracy                           0.95       231
   macro avg       0.95      0.95      0.95       231
weighted avg       0.95      0.95      0.95       231

  VGG16 + adam - Loss: 0.1696, Accuracy: 0.9524
```

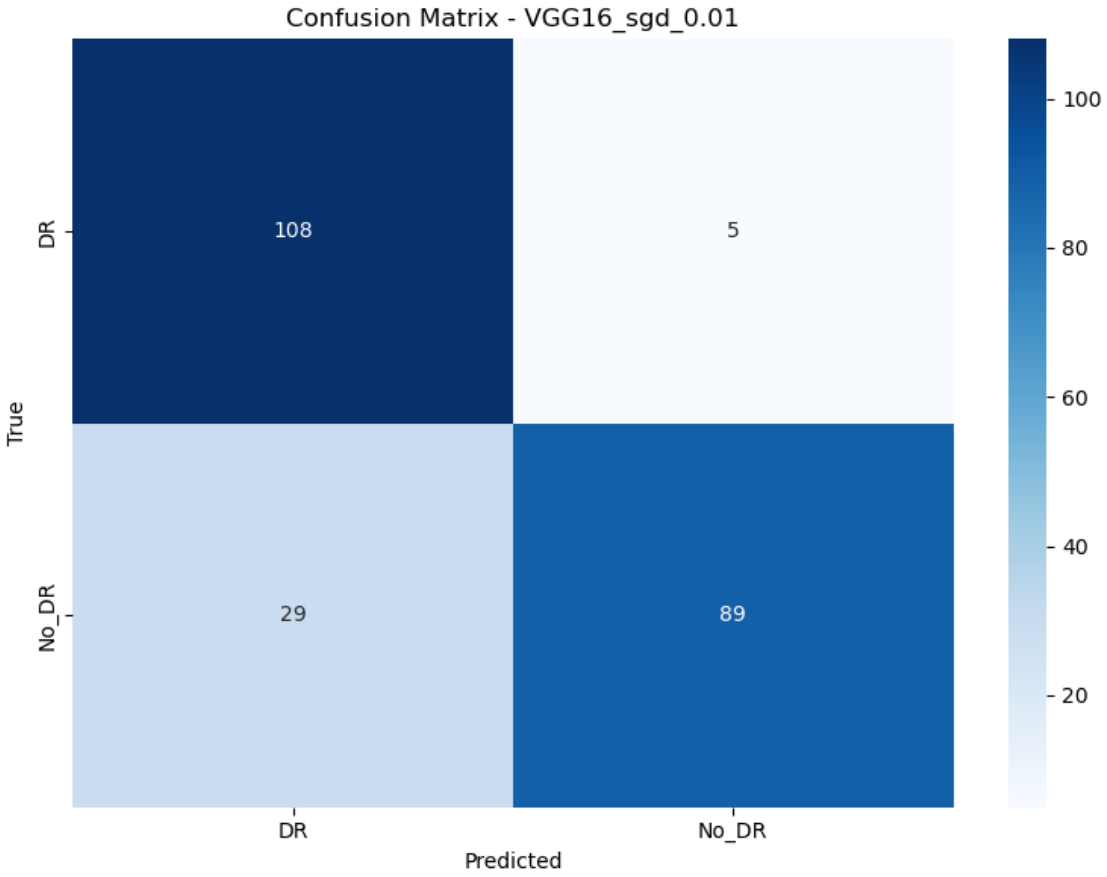
```
===== Training VGG16 with ADAM (lr=0.01) =====
8/8             45s 6s/step
```



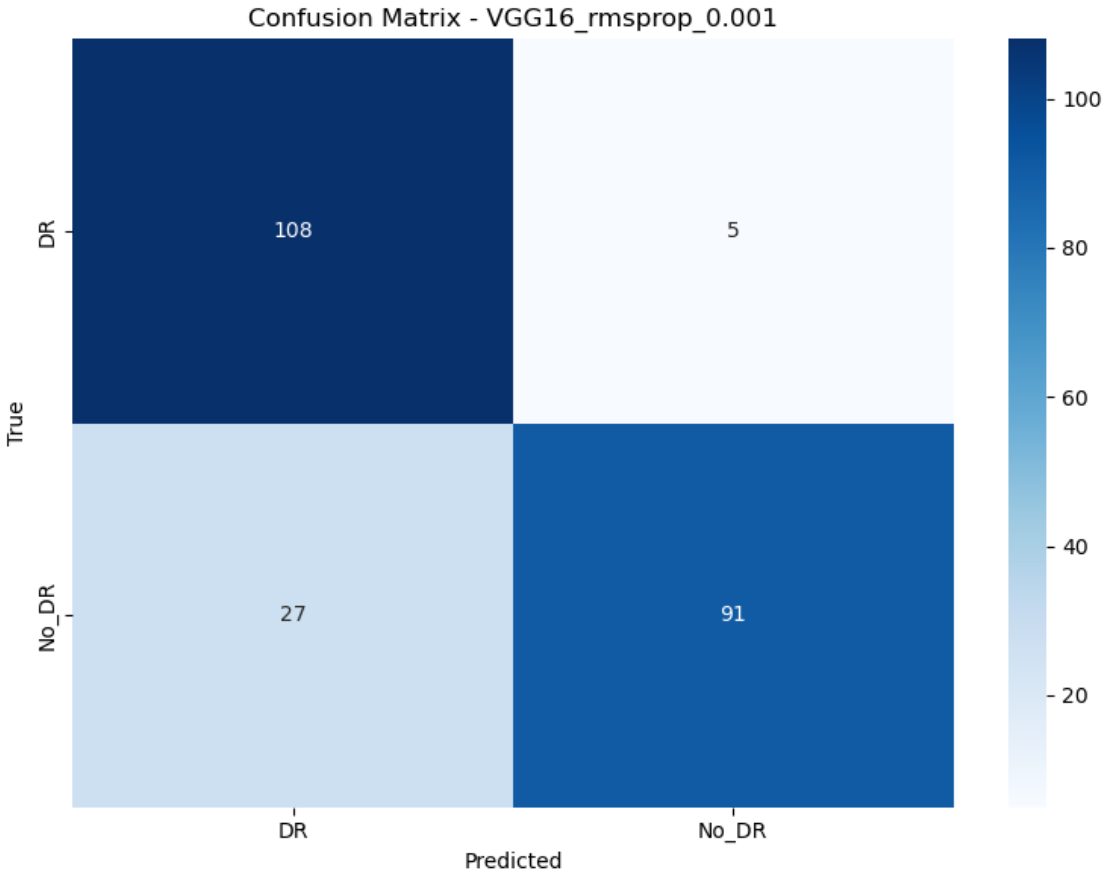

```
Classification Report - VGG16_adam_0.01:
              precision    recall  f1-score   support

          DR       0.91      0.88      0.90       113
       No_DR       0.89      0.92      0.90       118

    accuracy                           0.90       231
   macro avg       0.90      0.90      0.90       231
weighted avg       0.90      0.90      0.90       231

  VGG16 + adam - Loss: 0.2304, Accuracy: 0.9004
===== Training VGG16 with SGD (lr=0.001) =====
8/8             54s 6s/step
```



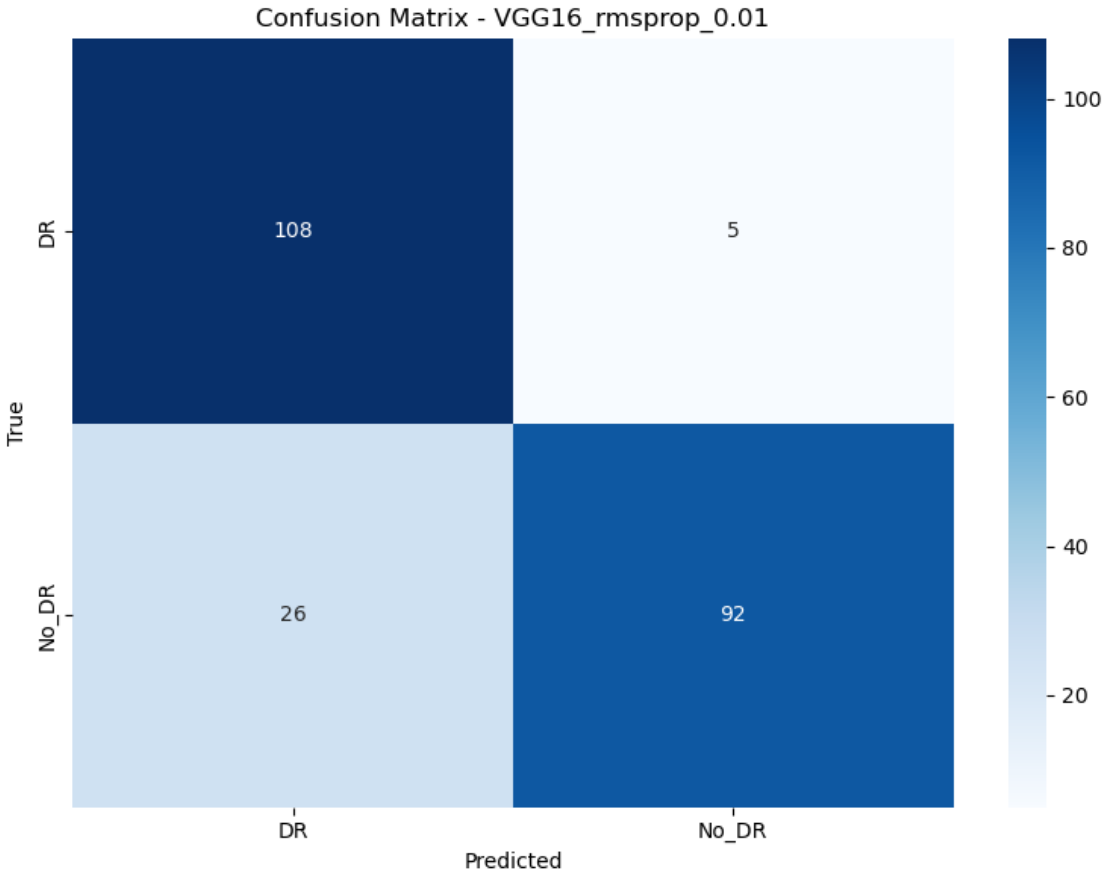

```
Classification Report - VGG16_sgd_0.001:
              precision    recall  f1-score   support

          DR       0.90      0.95      0.92       113
       No_DR       0.95      0.90      0.92       118

    accuracy                           0.92       231
   macro avg       0.92      0.92      0.92       231
weighted avg       0.92      0.92      0.92       231

  VGG16 + sgd - Loss: 0.2525, Accuracy: 0.9221
===== Training VGG16 with SGD (lr=0.01) =====
8/8             47s 6s/step
```



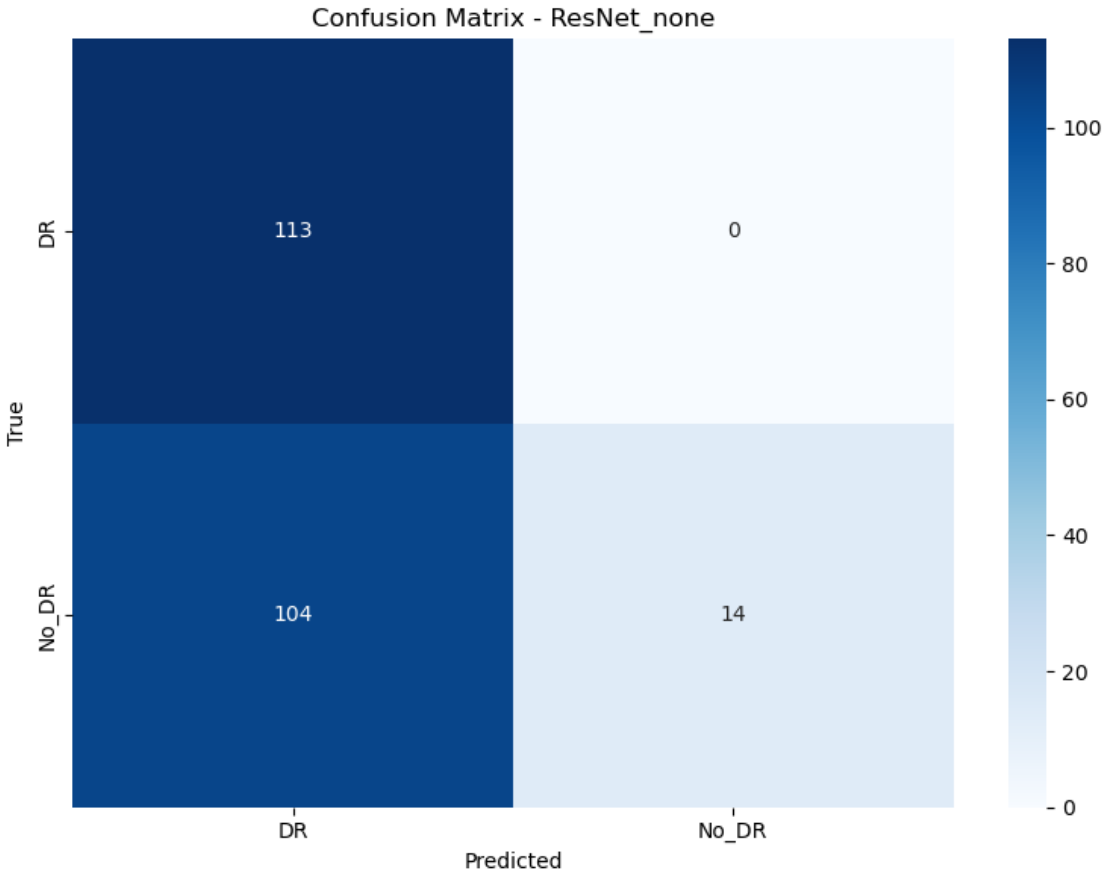

```
Classification Report - VGG16_sgd_0.01:
              precision    recall  f1-score   support

          DR       0.79      0.96      0.86       113
       No_DR       0.95      0.75      0.84       118

    accuracy                           0.85       231
   macro avg       0.87      0.85      0.85       231
weighted avg       0.87      0.85      0.85       231

  VGG16 + sgd - Loss: 0.2891, Accuracy: 0.8528
===== Training VGG16 with RMSPROP (lr=0.001) =====
8/8             48s 6s/step
```



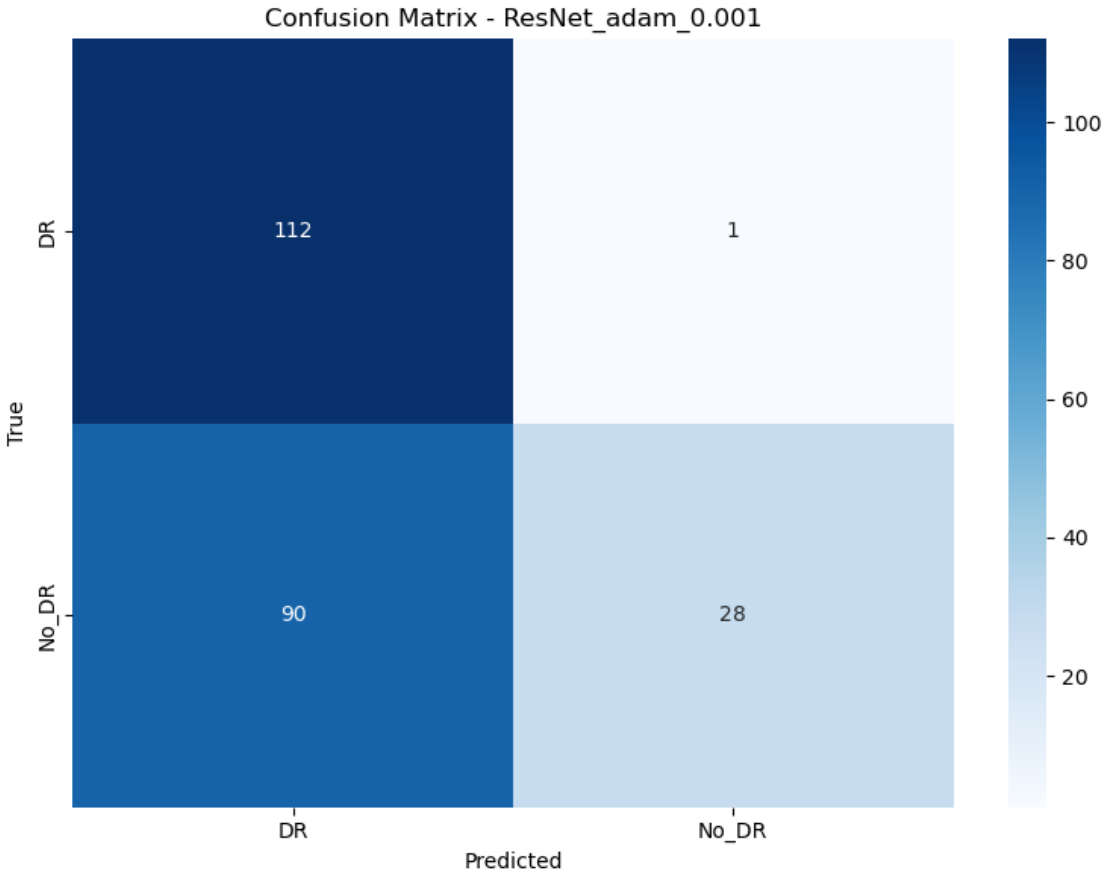

```
Classification Report - VGG16_rmsprop_0.001:
              precision    recall  f1-score   support

          DR       0.80      0.96      0.87       113
       No_DR       0.95      0.77      0.85       118

    accuracy                           0.86       231
   macro avg       0.87      0.86      0.86       231
weighted avg       0.88      0.86      0.86       231

  VGG16 + rmsprop - Loss: 0.3024, Accuracy: 0.8615
===== Training VGG16 with RMSPROP (lr=0.01) =====
8/8             46s 6s/step
```



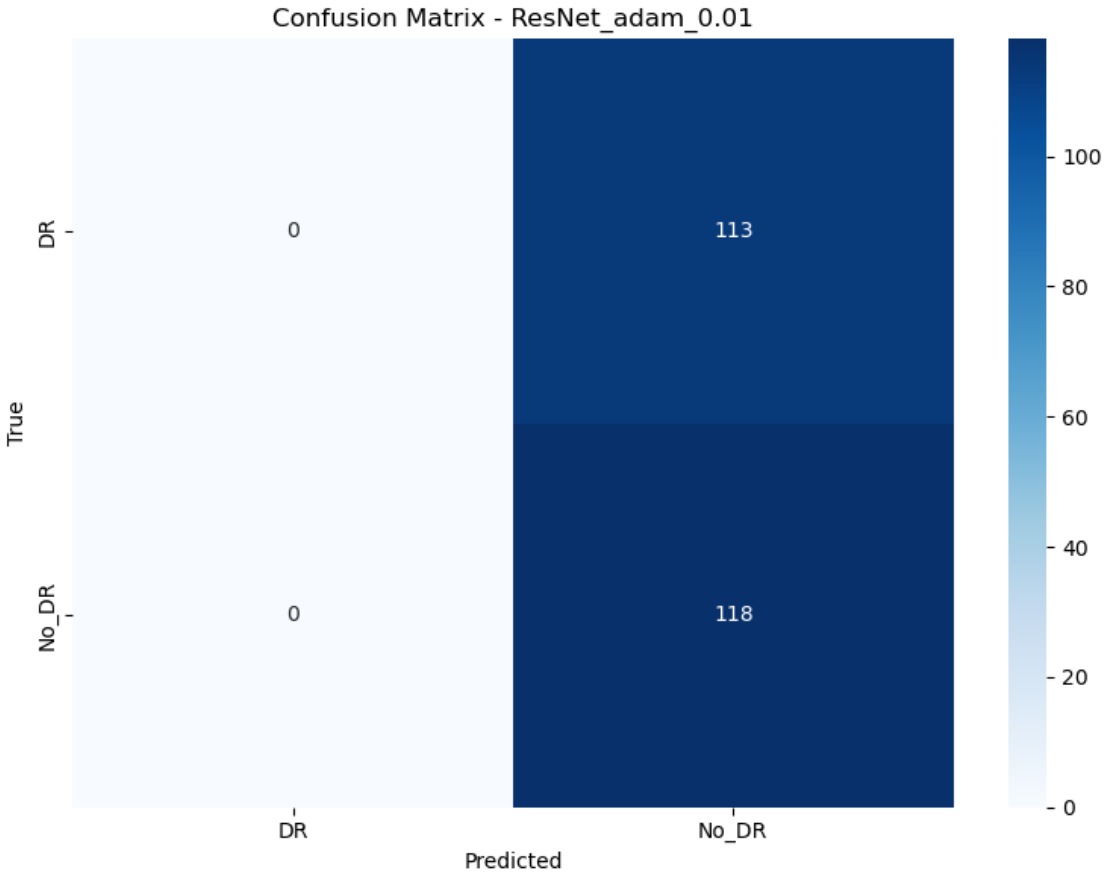

```
Classification Report - VGG16_rmsprop_0.01:
              precision    recall  f1-score   support

          DR       0.81      0.96      0.87       113
       No_DR       0.95      0.78      0.86       118

    accuracy                           0.87       231
   macro avg       0.88      0.87      0.87       231
weighted avg       0.88      0.87      0.86       231

  VGG16 + rmsprop - Loss: 0.2587, Accuracy: 0.8658
===== Training ResNet WITHOUT Optimizer =====
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736             87s
1us/step
8/8             25s 3s/step
```




```
Classification Report - ResNet_none:
              precision    recall  f1-score   support

          DR       0.52      1.00      0.68       113
       No_DR       1.00      0.12      0.21       118

    accuracy                           0.55       231
   macro avg       0.76      0.56      0.45       231
weighted avg       0.77      0.55      0.44       231

  ResNet no optimizer - Loss: 0.6646, Accuracy: 0.5498
===== Training ResNet with ADAM (lr=0.001) =====
8/8             21s 2s/step
```

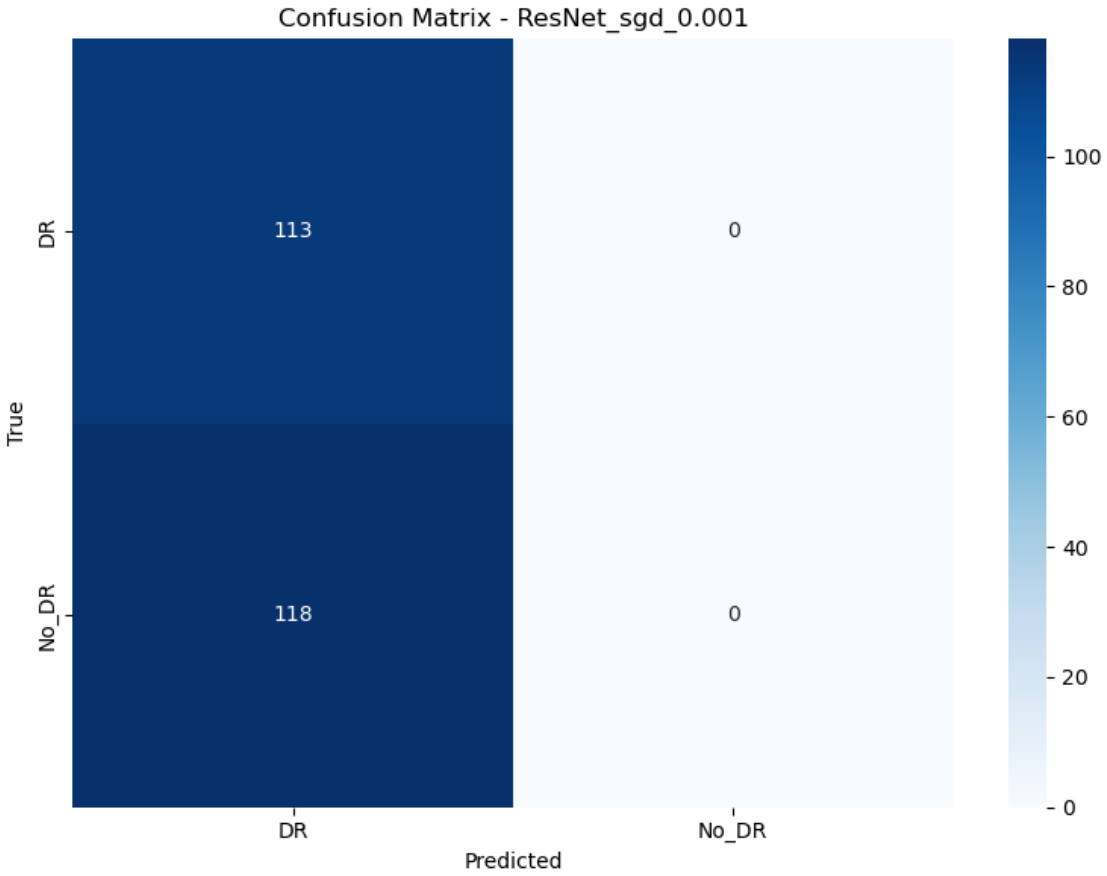



```
Classification Report - ResNet_adam_0.001:
              precision    recall  f1-score   support

          DR       0.55      0.99      0.71       113
       No_DR       0.97      0.24      0.38       118

    accuracy                           0.61       231
   macro avg       0.76      0.61      0.55       231
weighted avg       0.76      0.61      0.54       231

  ResNet + adam - Loss: 0.5884, Accuracy: 0.6061
===== Training ResNet with ADAM (lr=0.01) =====
8/8             20s 2s/step
```



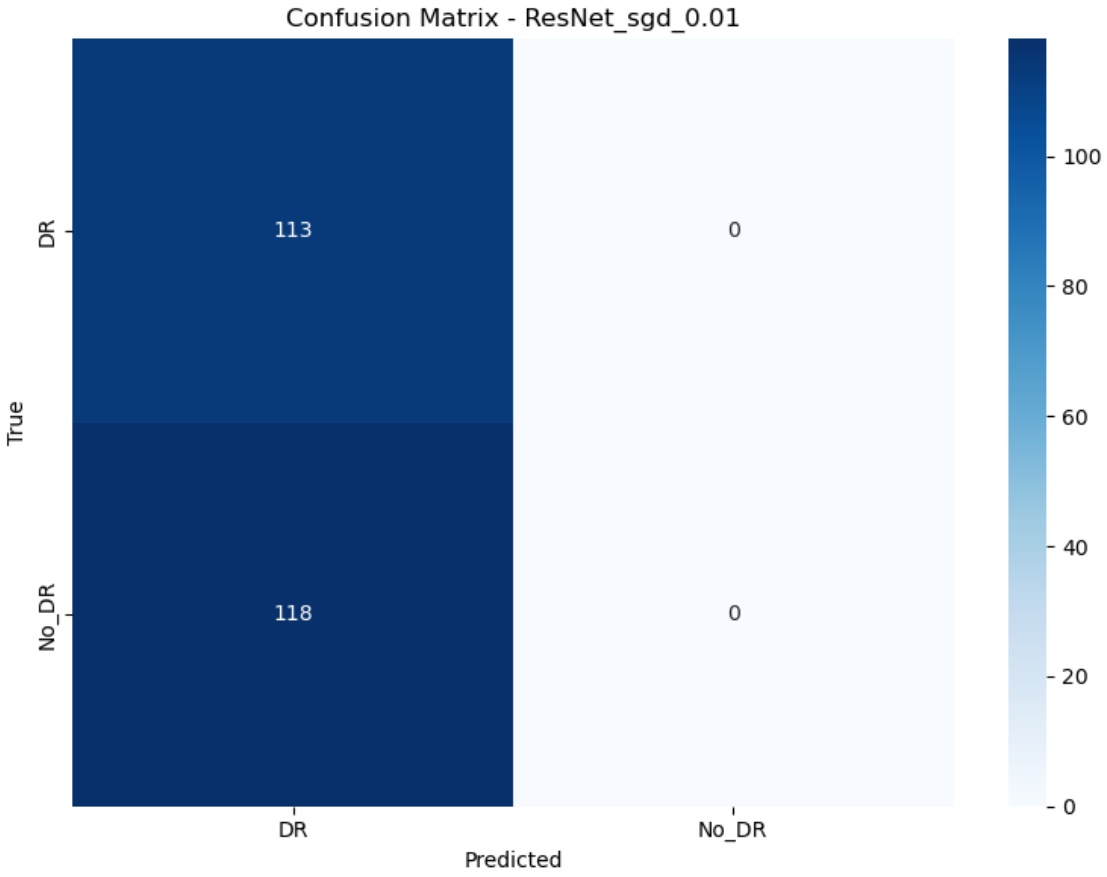

```
Classification Report - ResNet_adam_0.01:
              precision    recall  f1-score   support

          DR       0.00      0.00      0.00       113
       No_DR       0.51      1.00      0.68       118

    accuracy                           0.51       231
   macro avg       0.26      0.50      0.34       231
weighted avg       0.26      0.51      0.35       231

  ResNet + adam - Loss: 0.6931, Accuracy: 0.5108
===== Training ResNet with SGD (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
```

```
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
8/8             18s 2s/step
```

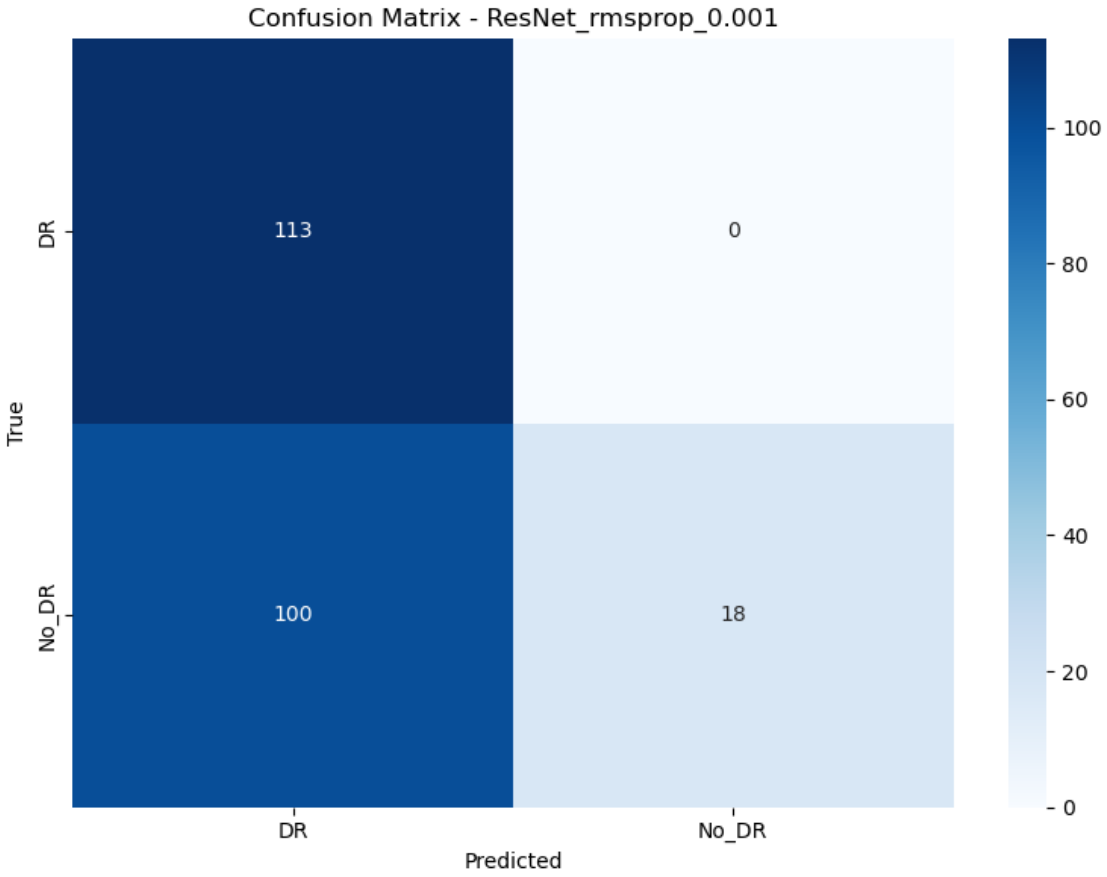



```
Classification Report - ResNet_sgd_0.001:
              precision    recall  f1-score   support

          DR       0.49      1.00      0.66       113
       No_DR       0.00      0.00      0.00       118

    accuracy                           0.49       231
   macro avg       0.24      0.50      0.33       231
weighted avg       0.24      0.49      0.32       231
```

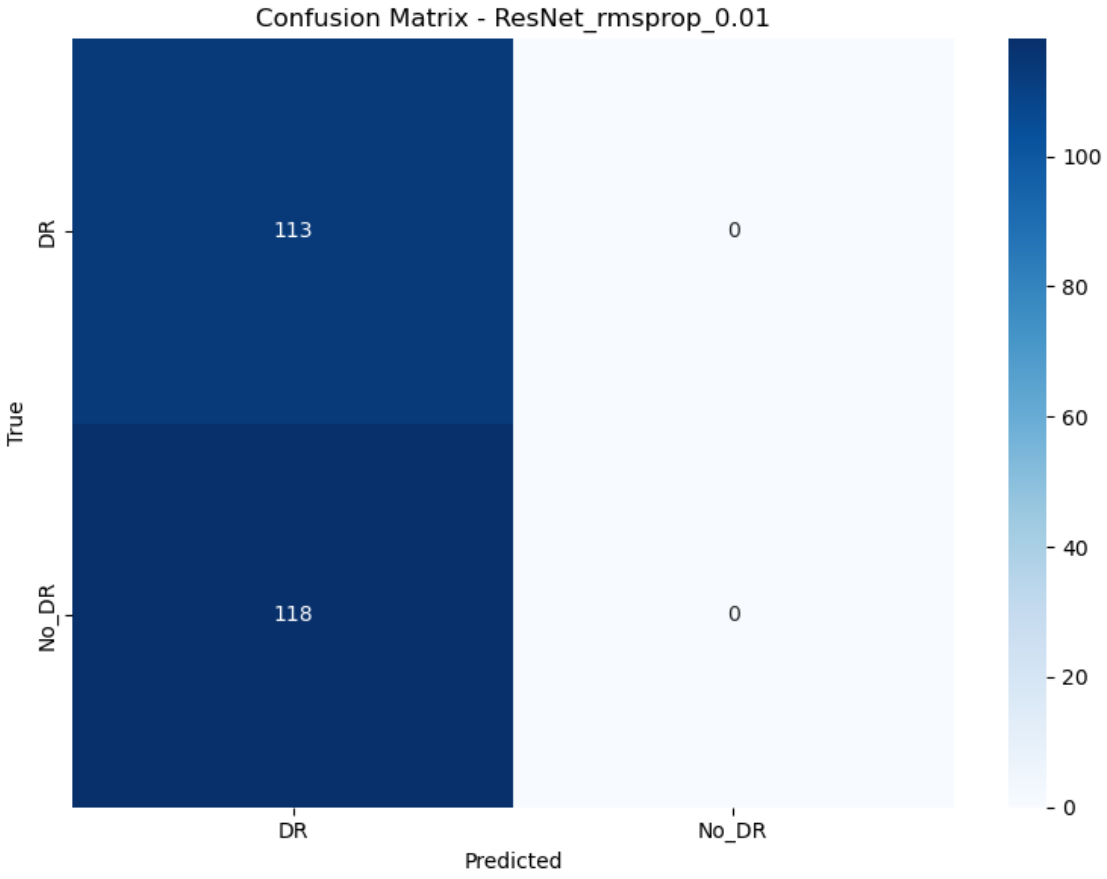
```
 ResNet + sgd - Loss: 0.6925, Accuracy: 0.4892
===== Training ResNet with SGD (lr=0.01) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

8/8             18s 2s/step
```



```
Classification Report - ResNet_sgd_0.01:
              precision    recall  f1-score   support

          DR       0.49      1.00      0.66       113
       No_DR       0.00      0.00      0.00       118

    accuracy                           0.49       231
   macro avg       0.24      0.50      0.33       231
weighted avg       0.24      0.49      0.32       231

  ResNet + sgd - Loss: 0.7254, Accuracy: 0.4892
===== Training ResNet with RMSPROP (lr=0.001) =====

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

8/8             17s 2s/step
```




```
Classification Report - ResNet_rmsprop_0.001:
              precision    recall  f1-score   support

          DR       0.53      1.00      0.69       113
       No_DR       1.00      0.15      0.26       118

    accuracy                           0.57       231
   macro avg       0.77      0.58      0.48       231
weighted avg       0.77      0.57      0.47       231

  ResNet + rmsprop - Loss: 0.6425, Accuracy: 0.5671
===== Training ResNet with RMSPROP (lr=0.01) =====
8/8             17s 2s/step
```

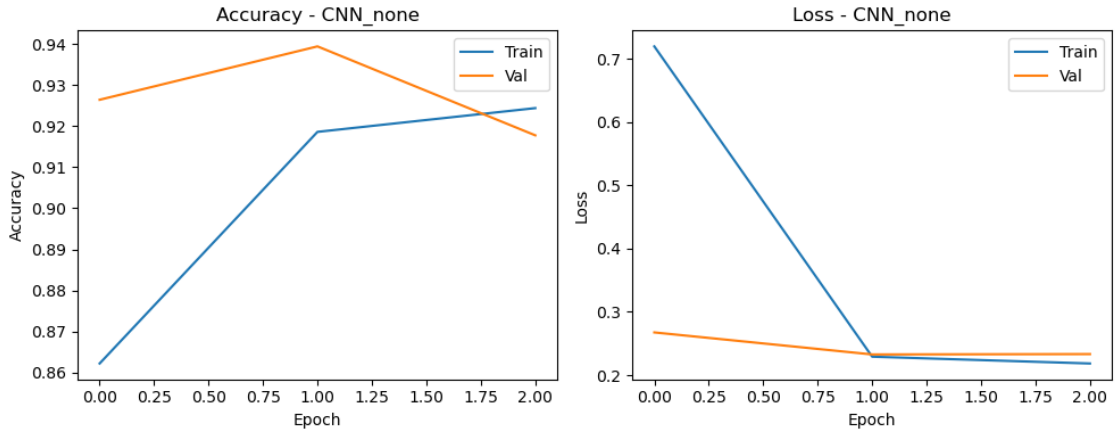


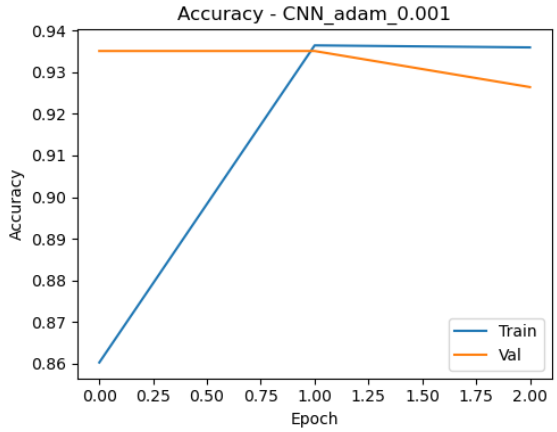

```
Classification Report - ResNet_rmsprop_0.01:
              precision    recall  f1-score   support

          DR       0.49      1.00      0.66       113
       No_DR       0.00      0.00      0.00       118

    accuracy                           0.49       231
   macro avg       0.24      0.50      0.33       231
weighted avg       0.24      0.49      0.32       231

  ResNet + rmsprop - Loss: 0.6932, Accuracy: 0.4892

C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
```

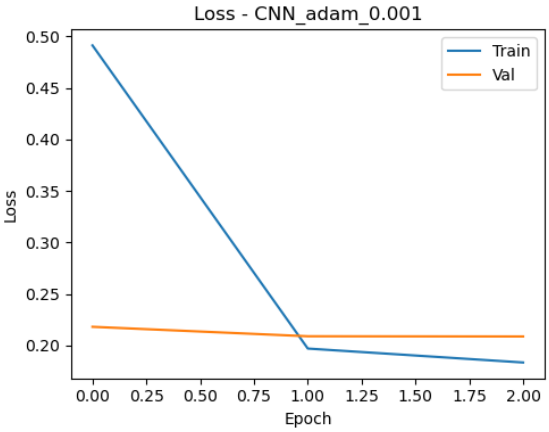
```
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\hp\.conda\envs\dnn\lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

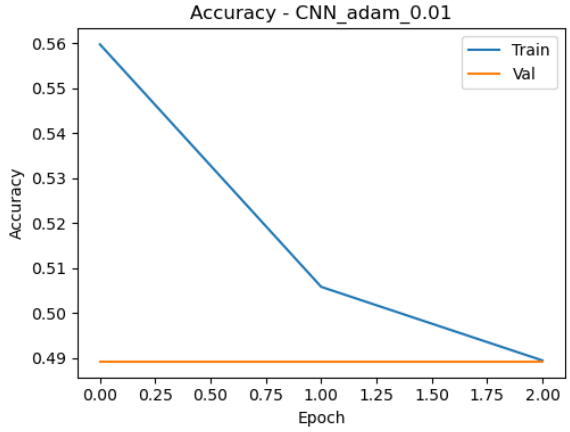
[18]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Plot individual Accuracy and Loss graphs per model
for key, history in history_logs.items():
    plt.figure(figsize=(10, 4))

    # Accuracy
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train')
    plt.plot(history.history['val_accuracy'], label='Val')
    plt.title(f'Accuracy - {key}')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    # Loss
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Train')
    plt.plot(history.history['val_loss'], label='Val')
    plt.title(f'Loss - {key}')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()

    plt.tight_layout()
    plt.show()

# Merged Accuracy & Loss Graphs
plt.figure(figsize=(12, 6))

# Merged Accuracy
for key, history in history_logs.items():
    plt.plot(history.history['val_accuracy'], label=f'{key}')
plt.title('Validation Accuracy Comparison')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
```

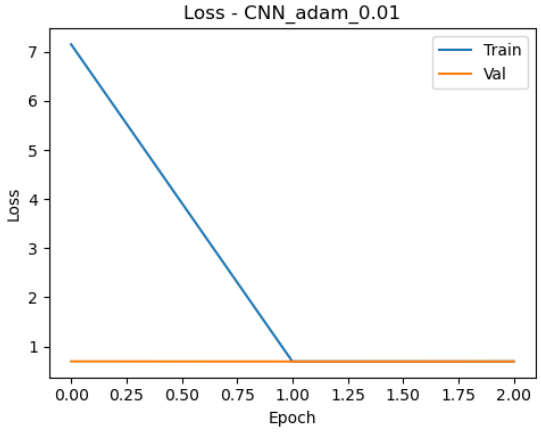
```
plt.grid(True)
plt.show()

# Merged Loss
plt.figure(figsize=(12, 6))
for key, history in history_logs.items():
    plt.plot(history.history['val_loss'], label=f'{key}')
plt.title('Validation Loss Comparison')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()

# Create summary report table
summary_df = pd.DataFrame(results, columns=['Model', 'Optimizer', 'Learning␣
 ↪Rate', 'Loss', 'Accuracy'])
summary_df = summary_df.sort_values(by='Accuracy', ascending=False)
summary_df.reset_index(drop=True, inplace=True)

print("\n Model Performance Summary:")
display(summary_df)

# Optional: Save to CSV
# summary_df.to_csv('model_summary_report.csv', index=False)
```

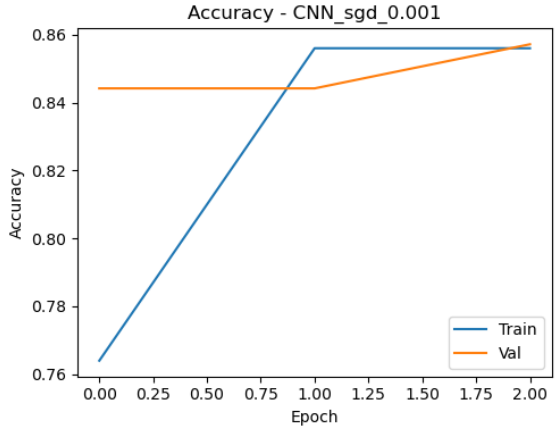


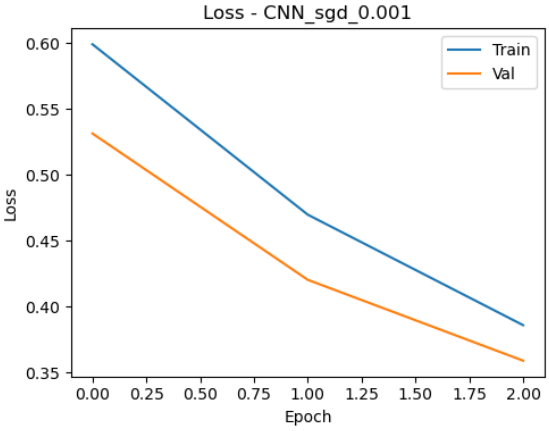

Accuracy - CNN_adam_0.001
Accuracy
Epoch
Train
Val

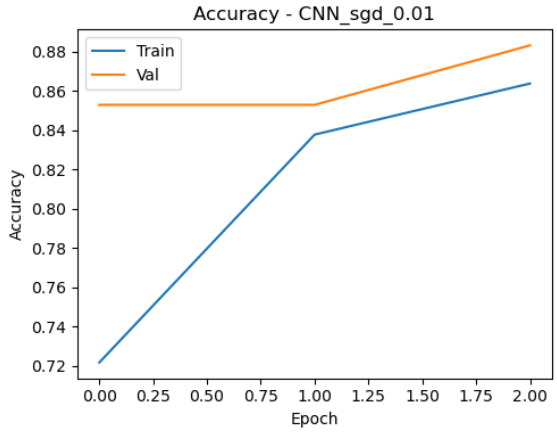

Loss - CNN_adam_0.001
Loss
Epoch
Train
Val

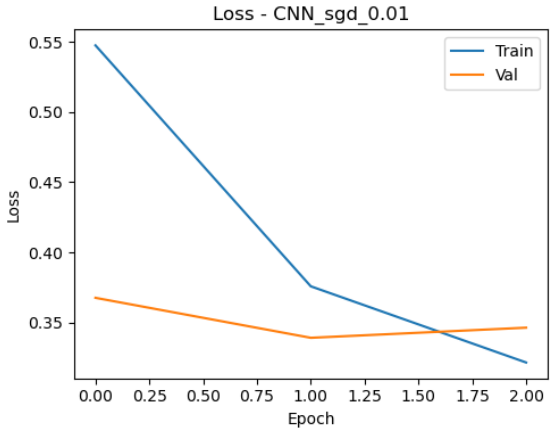

Accuracy - CNN_adam_0.01
Accuracy
Epoch
Train
Val

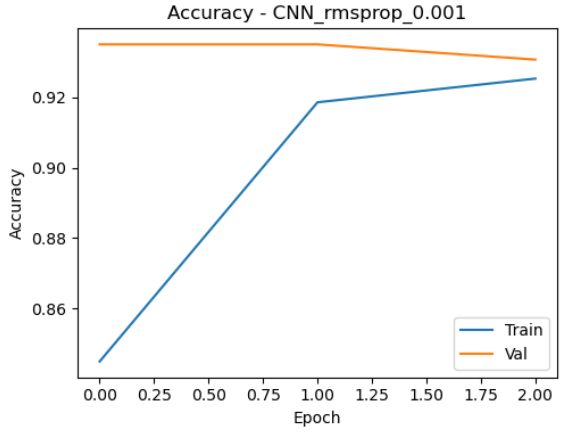

Loss - CNN_adam_0.01
Loss
Epoch
Train
Val

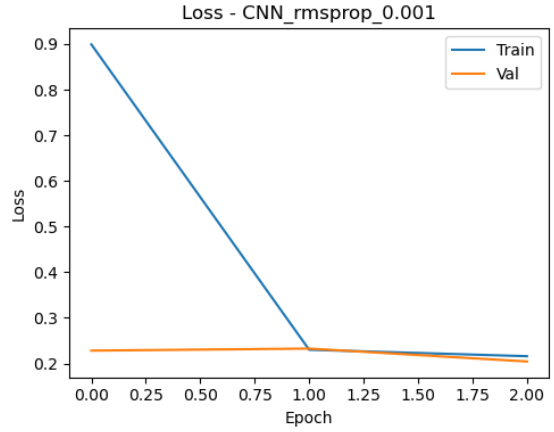

Accuracy - CNN_sgd_0.001
Accuracy
Epoch
Train
Val

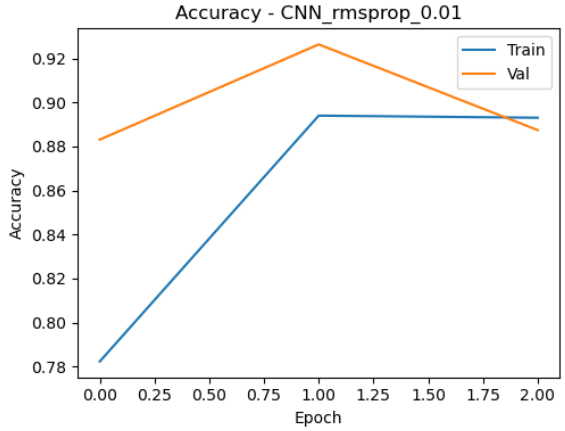

Loss - CNN_sgd_0.001
Loss
Epoch
Train
Val

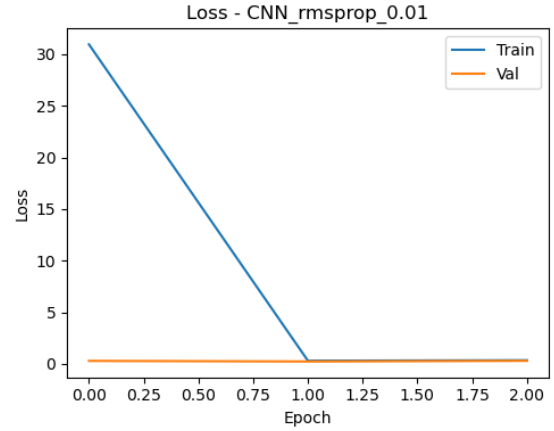

Accuracy - CNN_sgd_0.01
Train
Val
Accuracy
Epoch

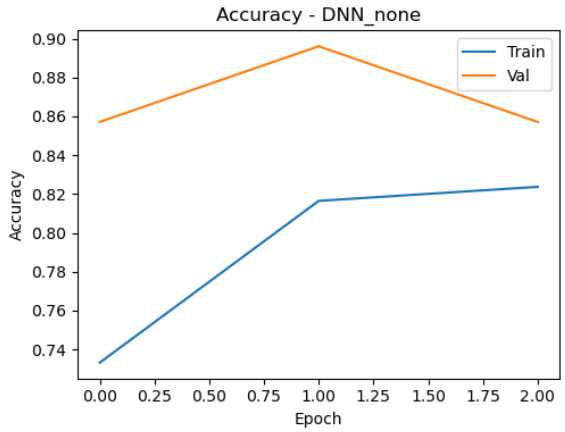

Loss - CNN_sgd_0.01
Train
Val
Loss
Epoch

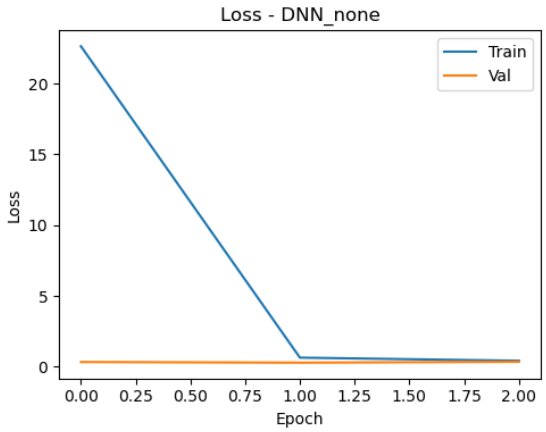

Accuracy - CNN_rmsprop_0.001
Train
Val
Accuracy
Epoch

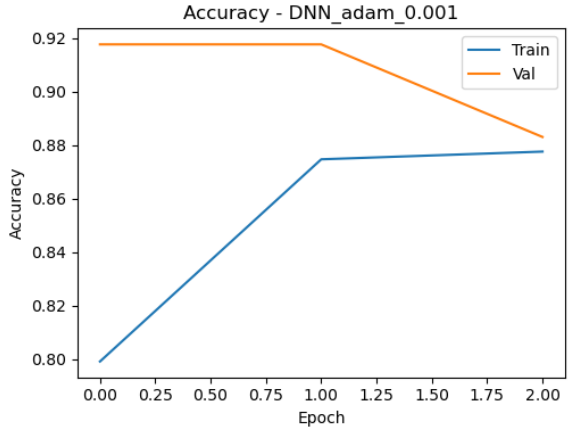

Loss - CNN_rmsprop_0.001
Train
Val
Loss
Epoch

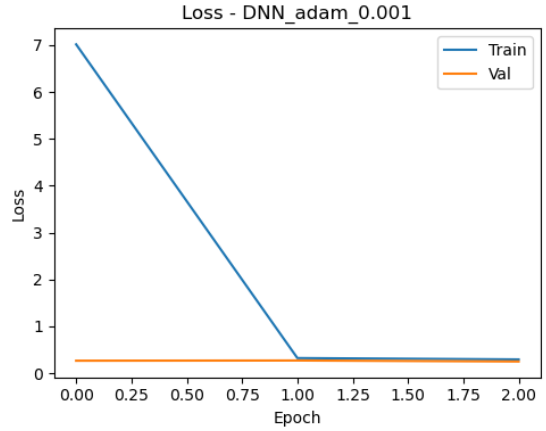

Accuracy - CNN_rmsprop_0.01
Train
Val
Accuracy
Epoch

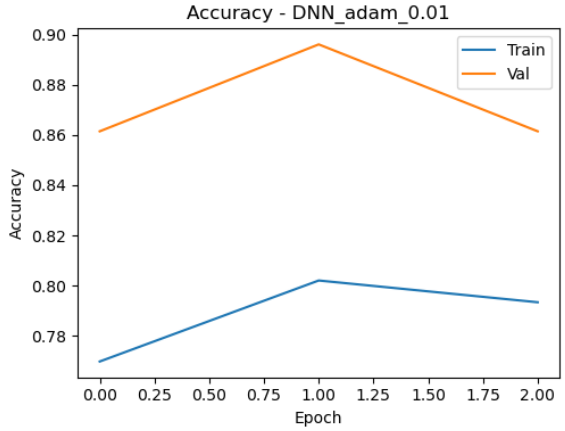

Loss - CNN_rmsprop_0.01
Train
Val
Loss
Epoch

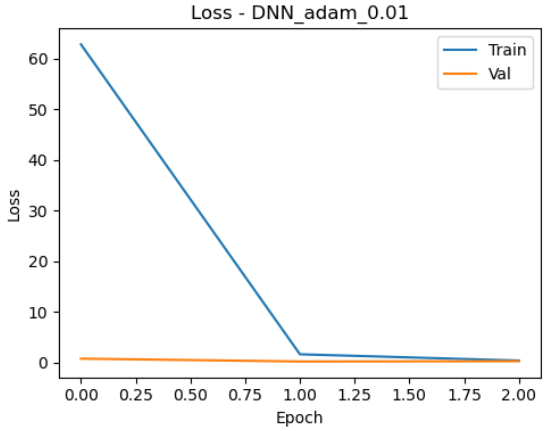

Accuracy - DNN_none
Train
Val
Accuracy
Epoch
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0.74
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

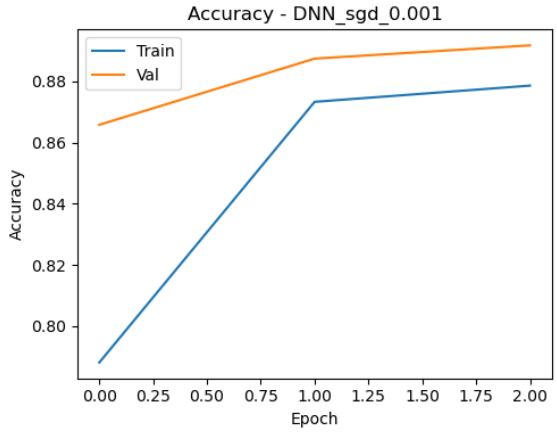

Loss - DNN_none
Train
Val
Loss
Epoch
20
15
10
5
0
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

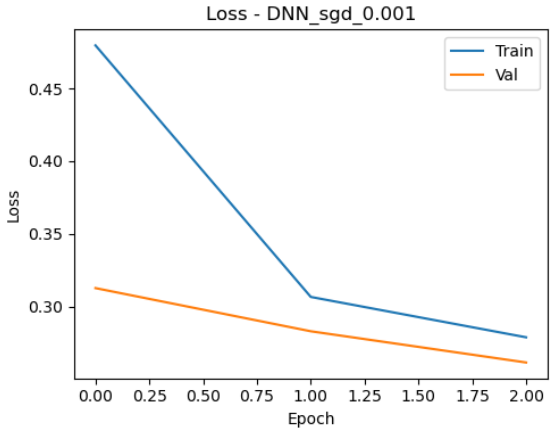

Accuracy - DNN_adam_0.001
Train
Val
Accuracy
Epoch
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

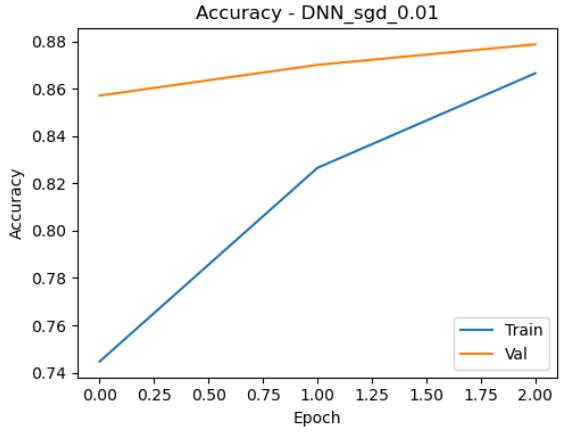

Loss - DNN_adam_0.001
Train
Val
Loss
Epoch
7
6
5
4
3
2
1
0
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

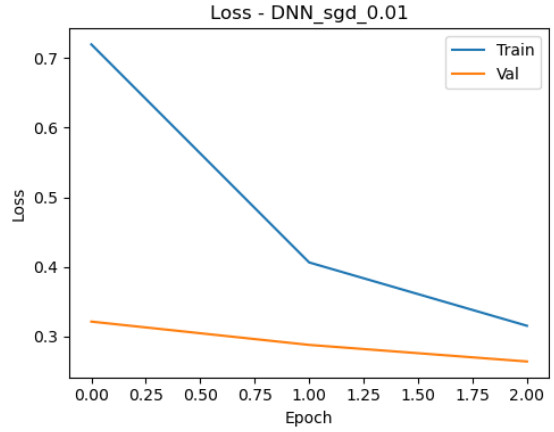

Accuracy - DNN_adam_0.01
Train
Val
Accuracy
Epoch
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

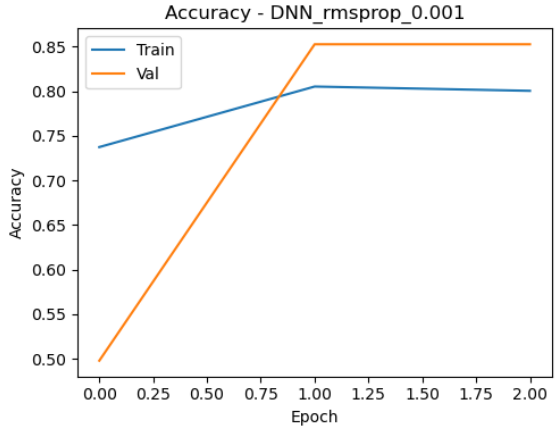

Loss - DNN_adam_0.01
Train
Val
Loss
Epoch
60
50
40
30
20
10
0
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

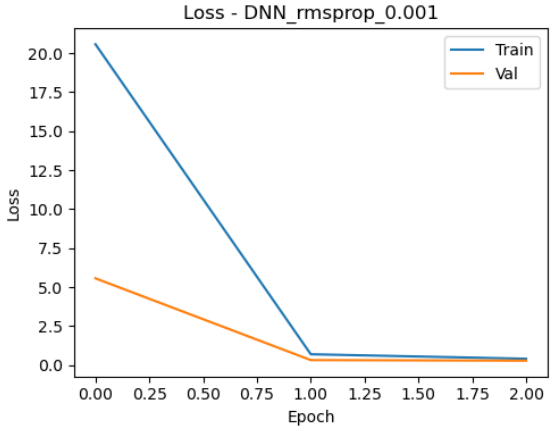

Accuracy - DNN_sgd_0.001
Train
Val
Accuracy
Epoch

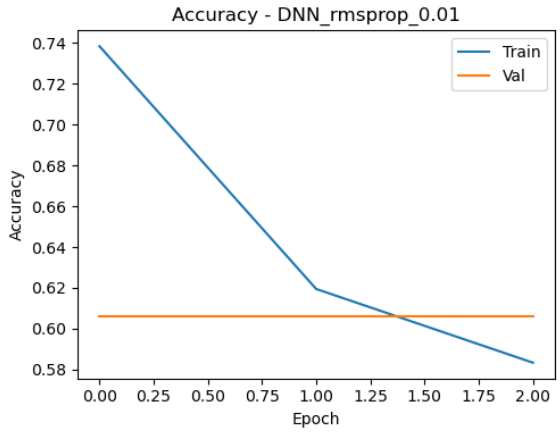

Loss - DNN_sgd_0.001
Train
Val
Loss
Epoch

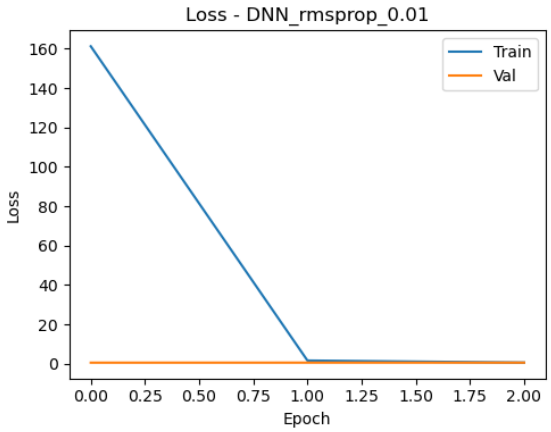

Accuracy - DNN_sgd_0.01
Train
Val
Accuracy
Epoch

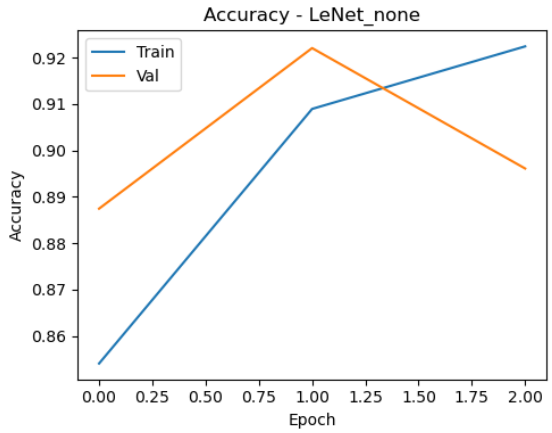

Loss - DNN_sgd_0.01
Train
Val
Loss
Epoch

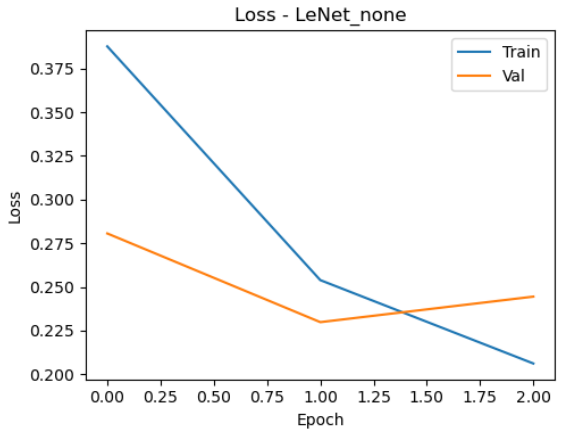

Accuracy - DNN_rmsprop_0.001
Train
Val
Accuracy
Epoch

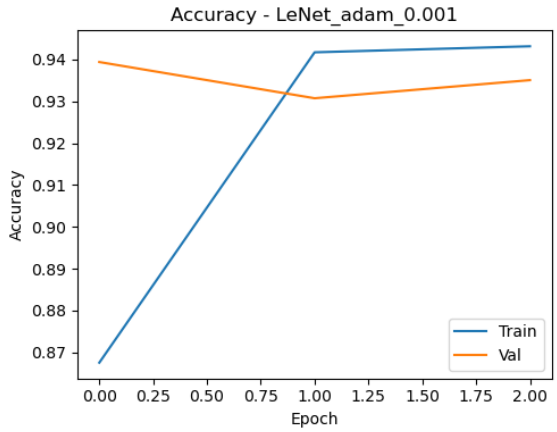

Loss - DNN_rmsprop_0.001
Train
Val
Loss
Epoch

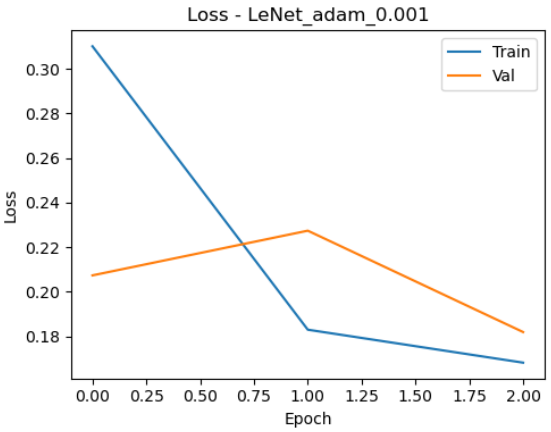

Accuracy - DNN_rmsprop_0.01
Train
Val
Accuracy
Epoch

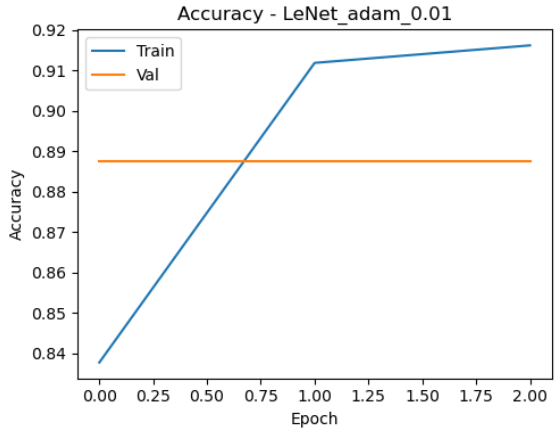

Loss - DNN_rmsprop_0.01
Train
Val
Loss
Epoch

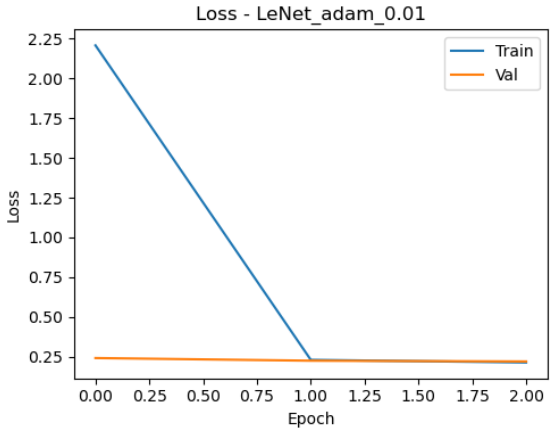

Accuracy - LeNet_none
Train
Val
Accuracy
Epoch

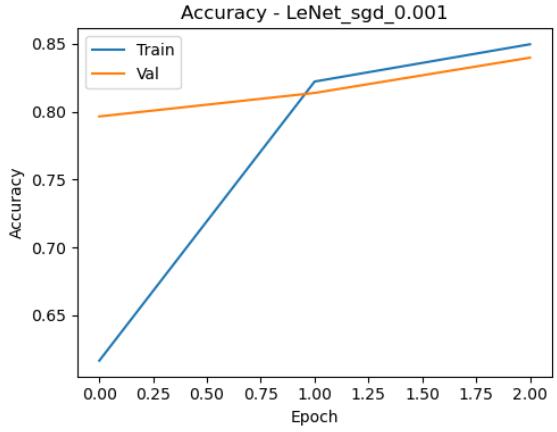

Loss - LeNet_none
Train
Val
Loss
Epoch

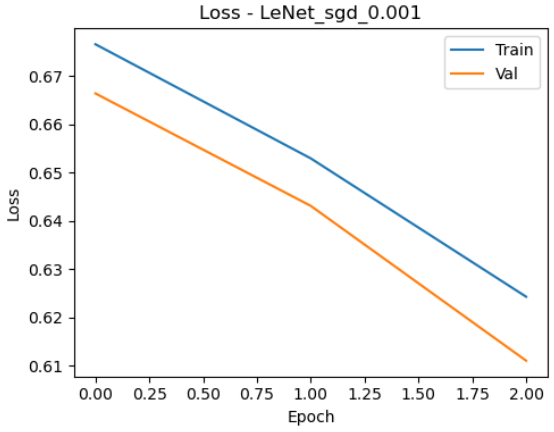

Accuracy - LeNet_adam_0.001
Train
Val
Accuracy
Epoch

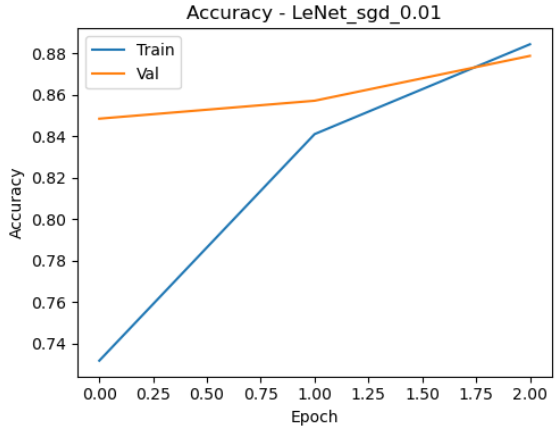

Loss - LeNet_adam_0.001
Train
Val
Loss
Epoch

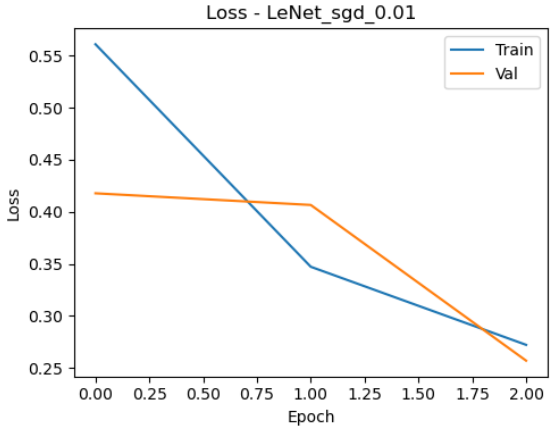

Accuracy - LeNet_adam_0.01
Train
Val
Accuracy
Epoch

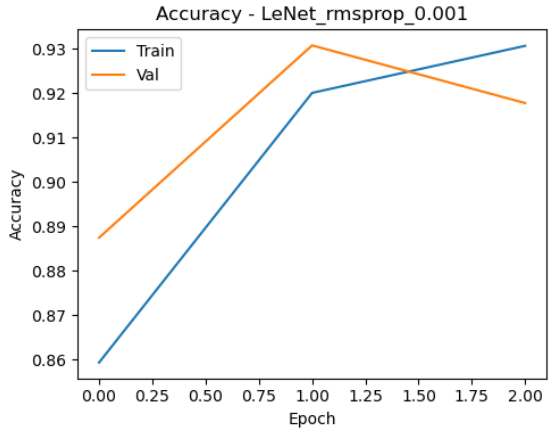

Loss - LeNet_adam_0.01
Train
Val
Loss
Epoch

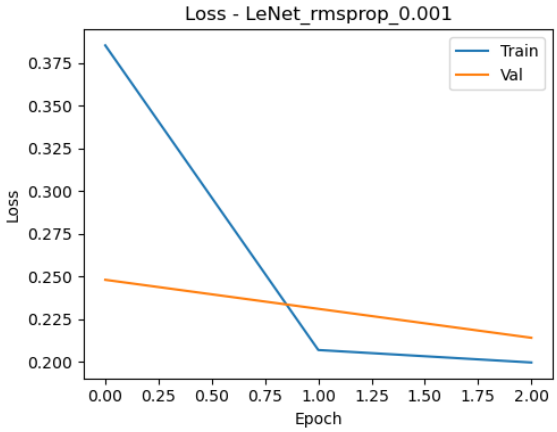

Accuracy - LeNet_sgd_0.001
Train
Val
Accuracy
Epoch

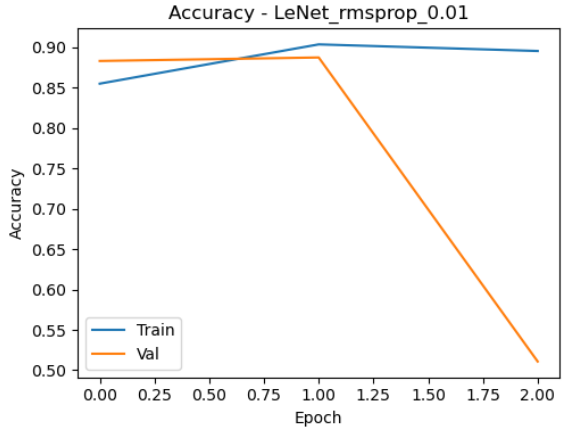

Loss - LeNet_sgd_0.001
Train
Val
Loss
Epoch

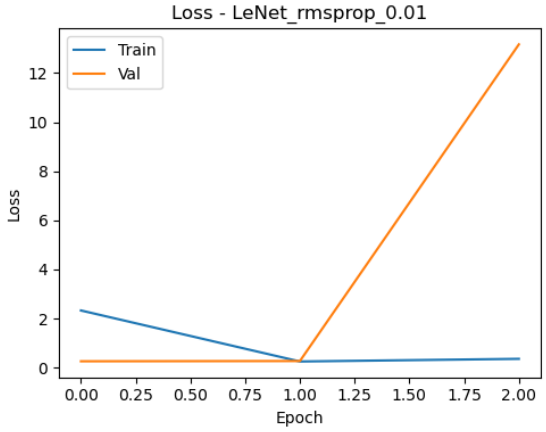

Accuracy - LeNet_sgd_0.01
Train
Val
Accuracy
Epoch

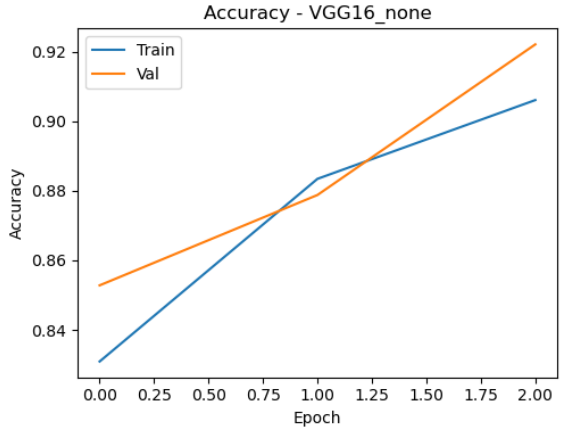

Loss - LeNet_sgd_0.01
Train
Val
Loss
Epoch

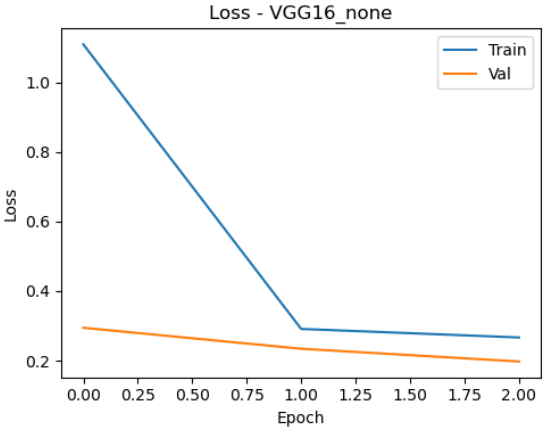

Accuracy - LeNet_rmsprop_0.001
Train
Val
Accuracy
Epoch

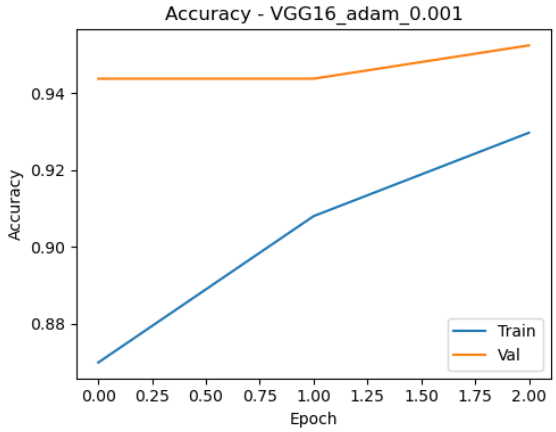

Loss - LeNet_rmsprop_0.001
Train
Val
Loss
Epoch

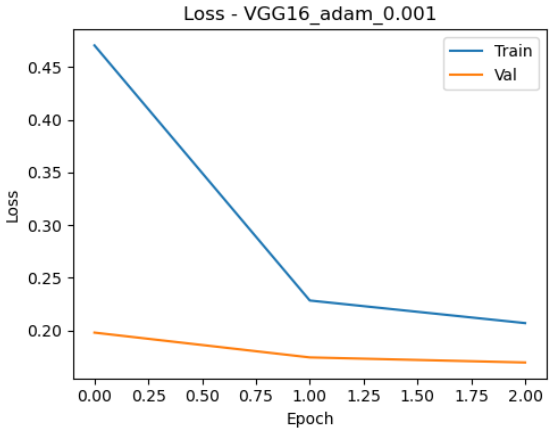

Accuracy - LeNet_rmsprop_0.01
Train
Val
Accuracy
Epoch

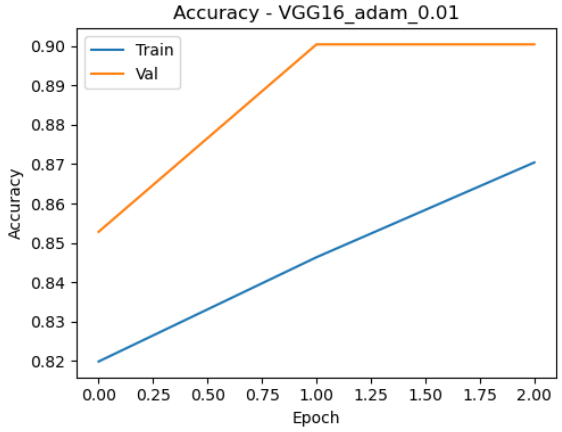

Loss - LeNet_rmsprop_0.01
Train
Val
Loss
Epoch

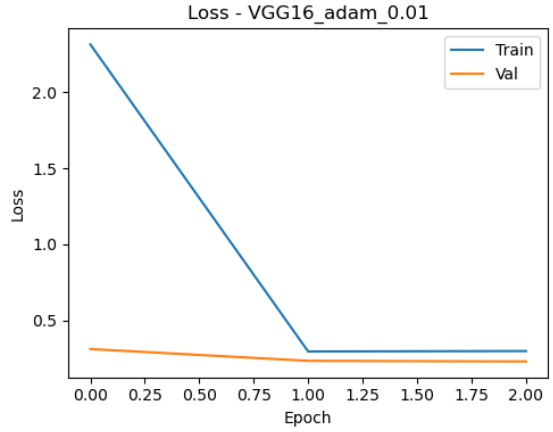

Accuracy - VGG16_none
Train
Val
Accuracy
Epoch

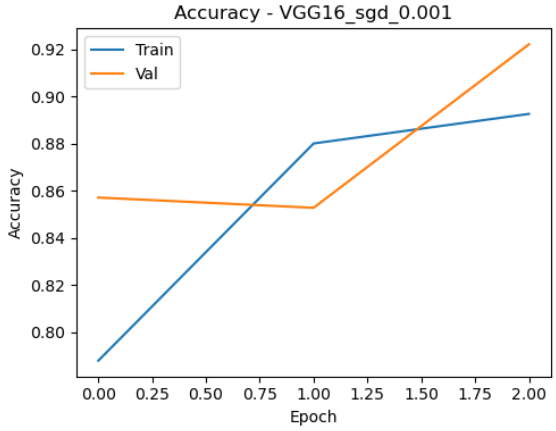

Loss - VGG16_none
Train
Val
Loss
Epoch

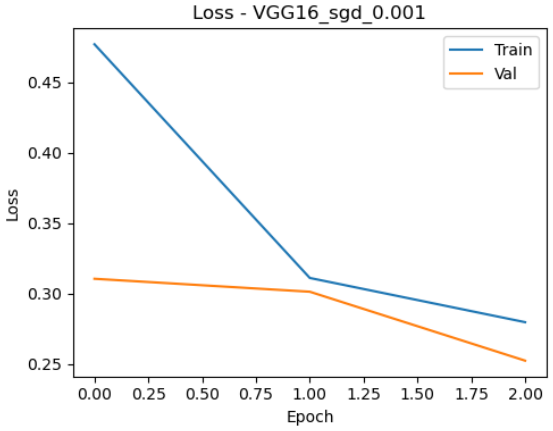

Accuracy - VGG16_adam_0.001
Accuracy
Epoch
Train
Val

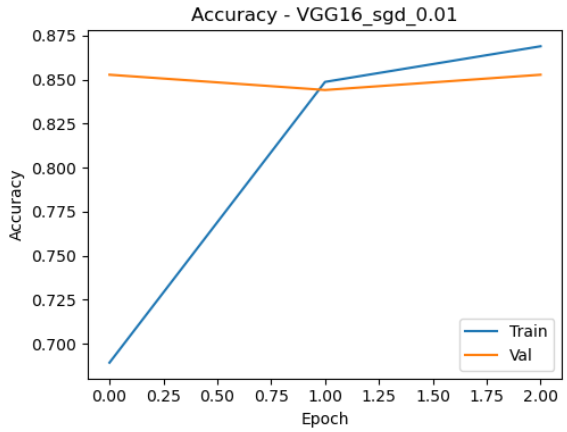

Loss - VGG16_adam_0.001
Loss
Epoch
Train
Val

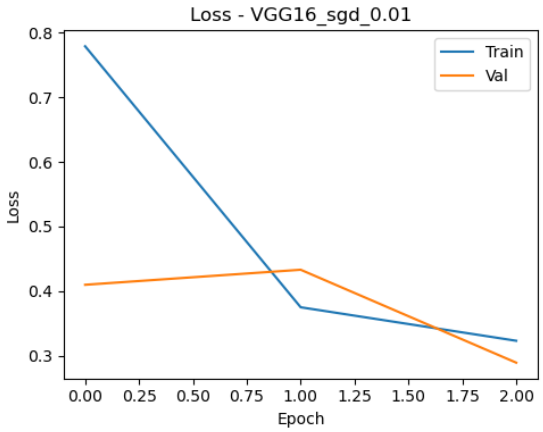

Accuracy - VGG16_adam_0.01
Accuracy
Epoch
Train
Val

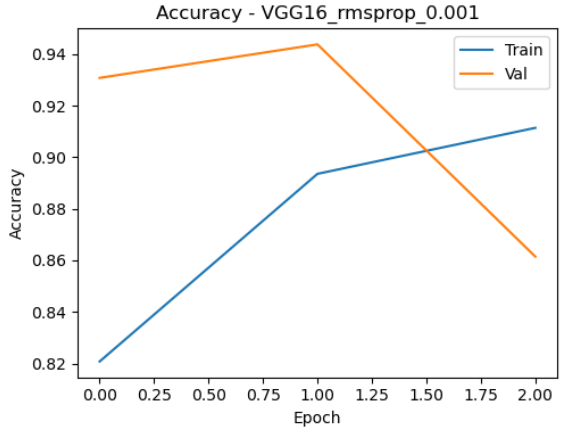

Loss - VGG16_adam_0.01
Loss
Epoch
Train
Val

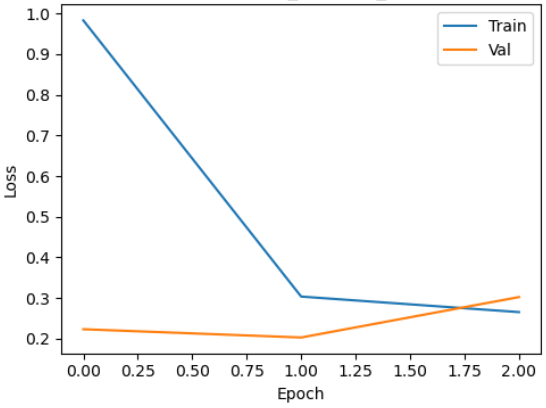

Accuracy - VGG16_sgd_0.001
Accuracy
Epoch
Train
Val

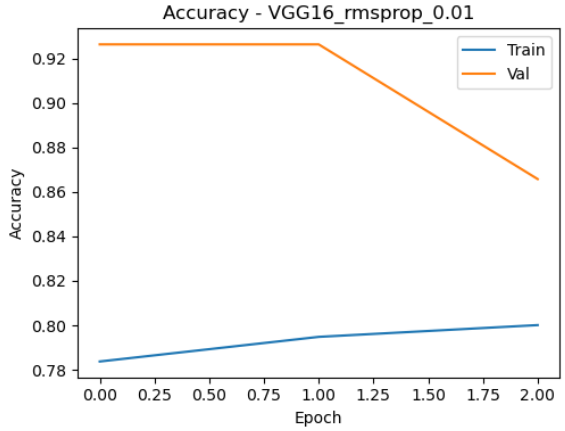

Loss - VGG16_sgd_0.001
Loss
Epoch
Train
Val

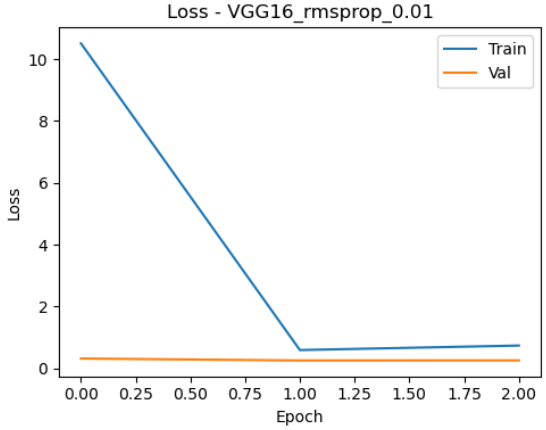

Accuracy - VGG16_sgd_0.01
Accuracy
Epoch
Train
Val

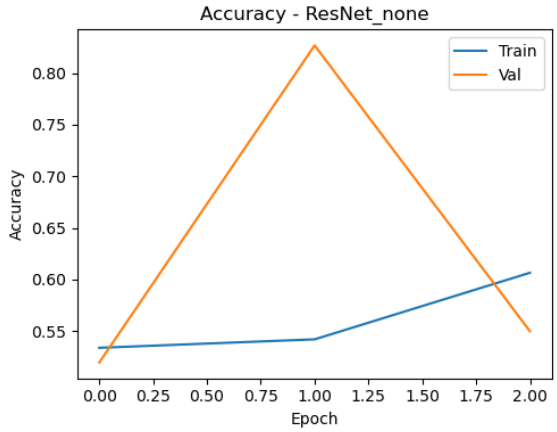

Loss - VGG16_sgd_0.01
Loss
Epoch
Train
Val

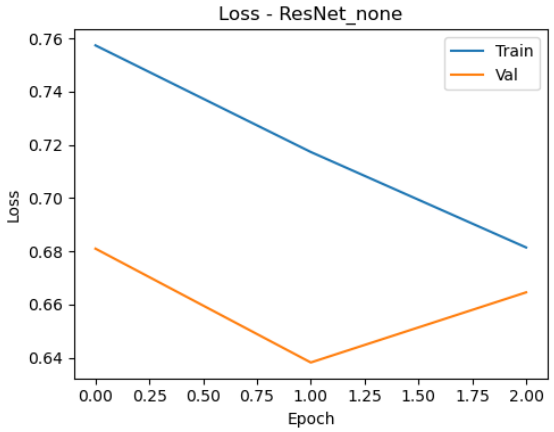

Accuracy - VGG16_rmsprop_0.001
Accuracy
Epoch
Train
Val

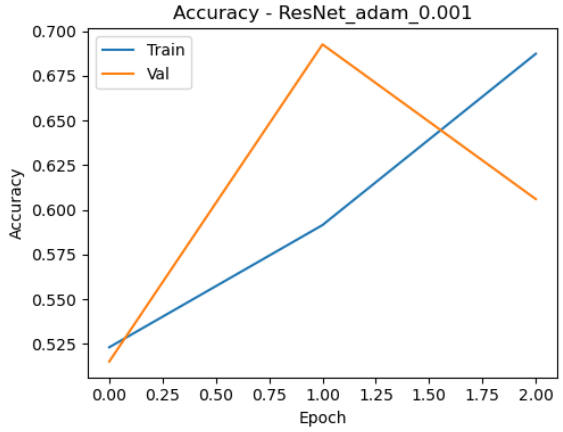

Loss - VGG16_rmsprop_0.001
Loss
Epoch
Train
Val

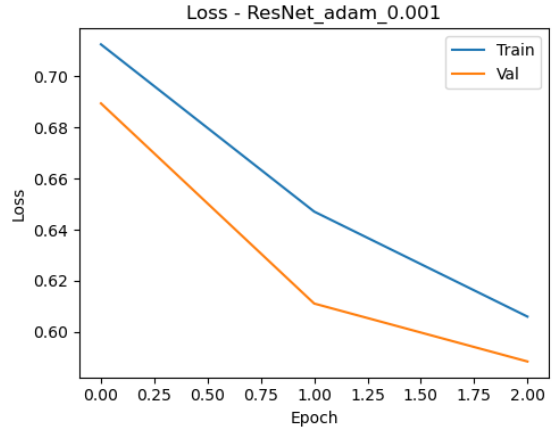

Accuracy - VGG16_rmsprop_0.01
Accuracy
Epoch
Train
Val

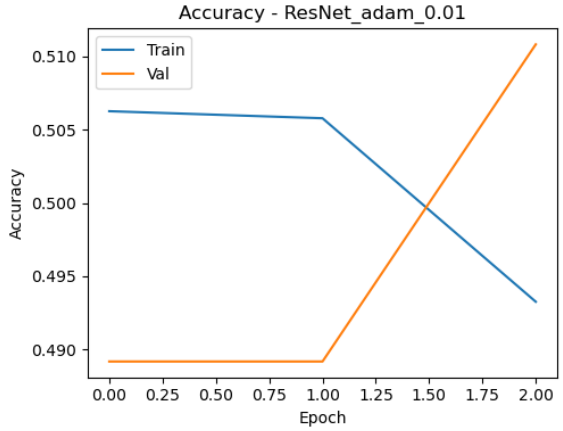

Loss - VGG16_rmsprop_0.01
Loss
Epoch
Train
Val

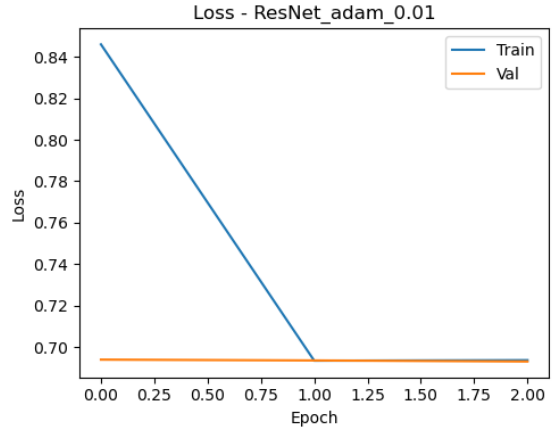

Accuracy - ResNet_none
Train
Val
Accuracy
Epoch

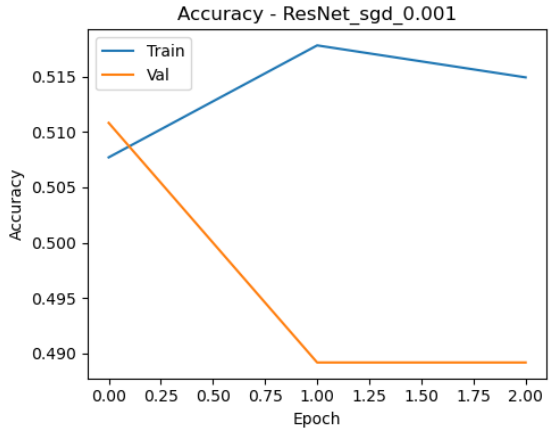

Loss - ResNet_none
Train
Val
Loss
Epoch

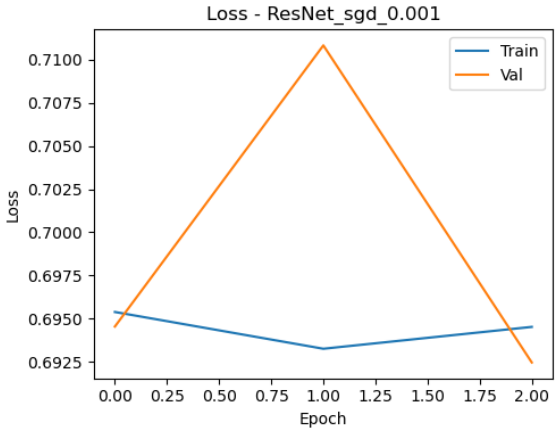

Accuracy - ResNet_adam_0.001
Train
Val
Accuracy
Epoch

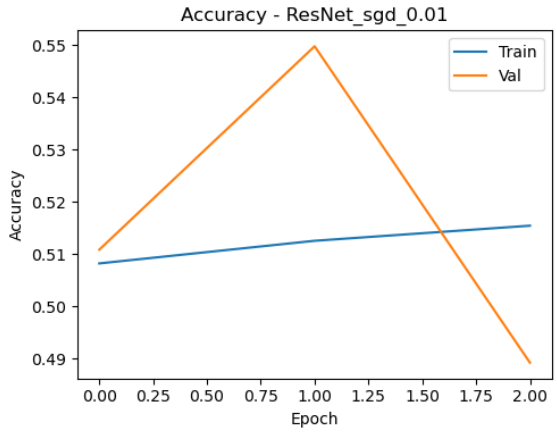

Loss - ResNet_adam_0.001
Train
Val
Loss
Epoch

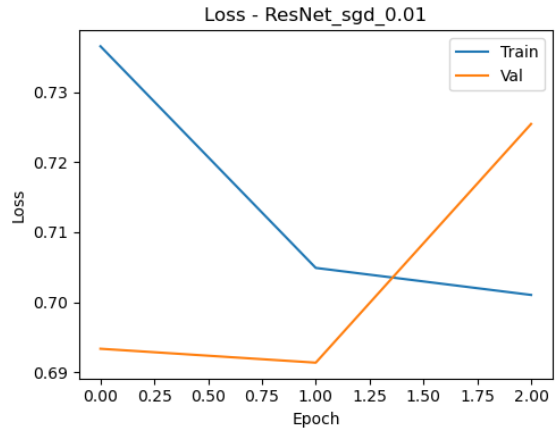

Accuracy - ResNet_adam_0.01
Train
Val
Accuracy
Epoch

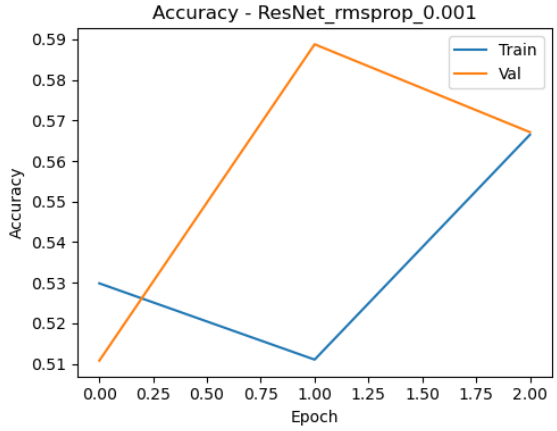

Loss - ResNet_adam_0.01
Train
Val
Loss
Epoch

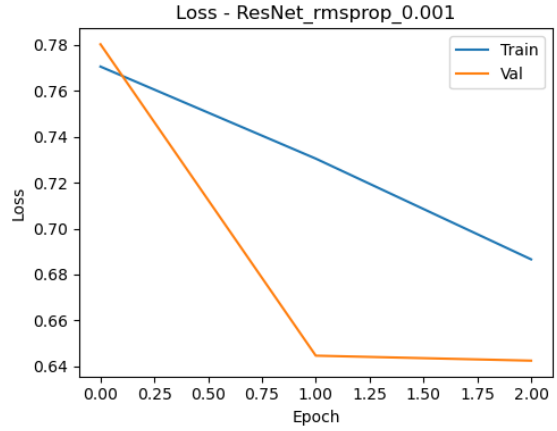

Accuracy - ResNet_sgd_0.001
Train
Val
Accuracy
Epoch

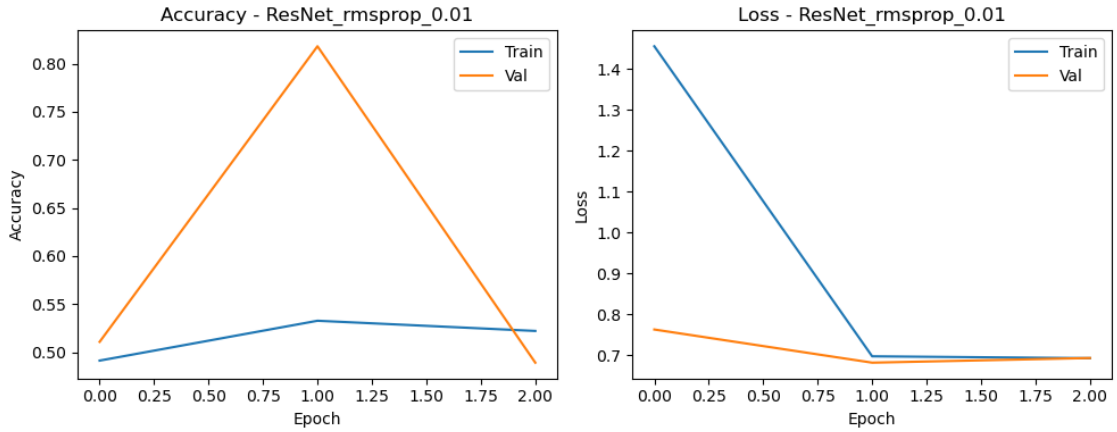

Loss - ResNet_sgd_0.001
Train
Val
Loss
Epoch

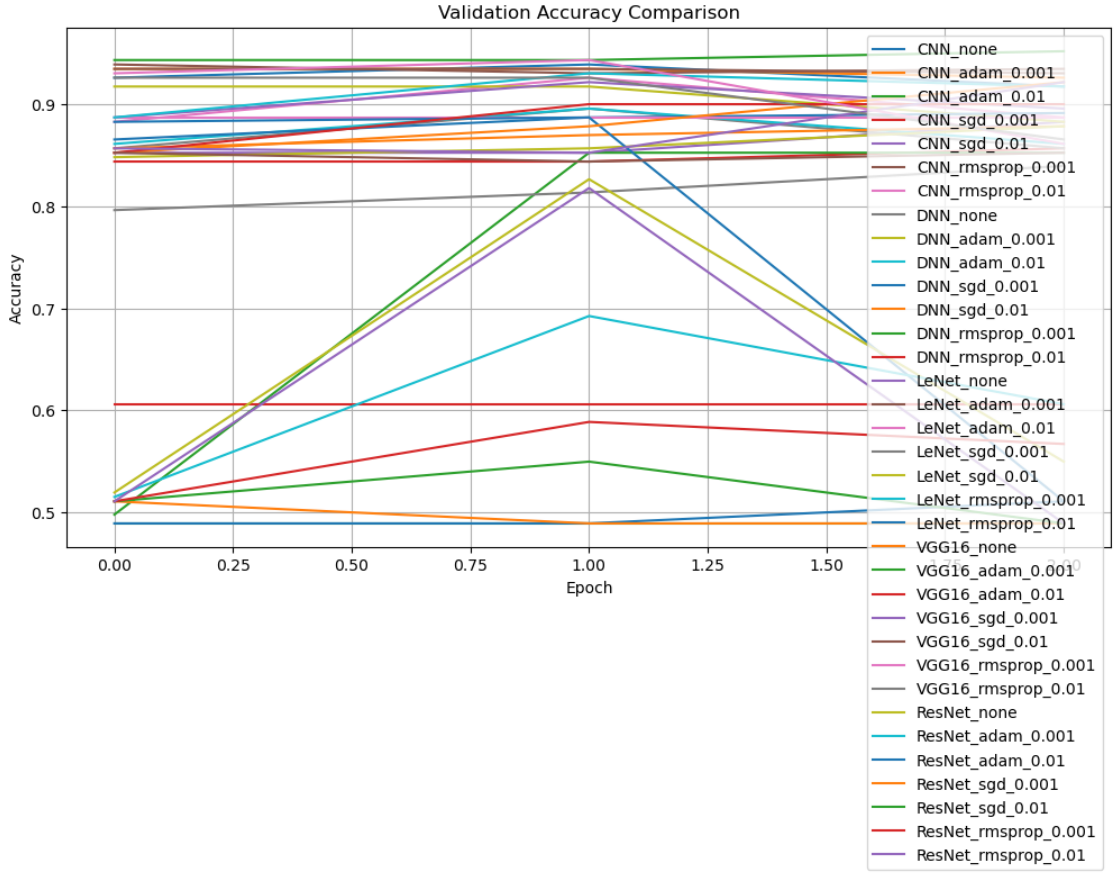

Accuracy - ResNet_sgd_0.01
Train
Val
Accuracy
Epoch

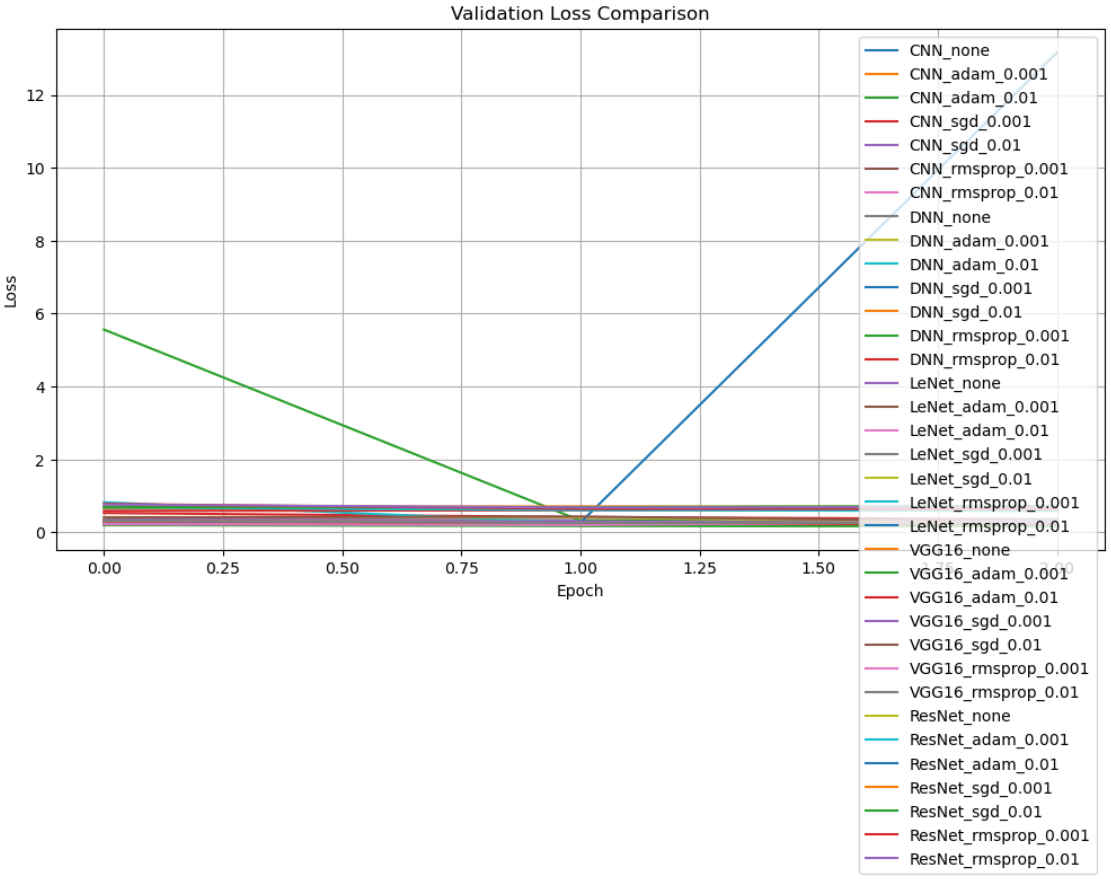

Loss - ResNet_sgd_0.01
Train
Val
Loss
Epoch


Accuracy - ResNet_rmsprop_0.001
Train
Val
Accuracy
Epoch


Loss - ResNet_rmsprop_0.001
Train
Val
Loss
Epoch


Accuracy - ResNet_rmsprop_0.01
Train
Val
Accuracy
Epoch
Loss - ResNet_rmsprop_0.01
Loss


Validation Accuracy Comparison
Accuracy
Epoch
CNN_none
CNN_adam_0.001
CNN_adam_0.01
CNN_sgd_0.001
CNN_sgd_0.01
CNN_rmsprop_0.001
CNN_rmsprop_0.01
DNN_none
DNN_adam_0.001
DNN_adam_0.01
DNN_sgd_0.001
DNN_sgd_0.01
DNN_rmsprop_0.001
DNN_rmsprop_0.01
LeNet_none
LeNet_adam_0.001
LeNet_adam_0.01
LeNet_sgd_0.001
LeNet_sgd_0.01
LeNet_rmsprop_0.001
LeNet_rmsprop_0.01
VGG16_none
VGG16_adam_0.001
VGG16_adam_0.01
VGG16_sgd_0.001
VGG16_sgd_0.01
VGG16_rmsprop_0.001
VGG16_rmsprop_0.01
ResNet_none
ResNet_adam_0.001
ResNet_adam_0.01
ResNet_sgd_0.001
ResNet_sgd_0.01
ResNet_rmsprop_0.001
ResNet_rmsprop_0.01



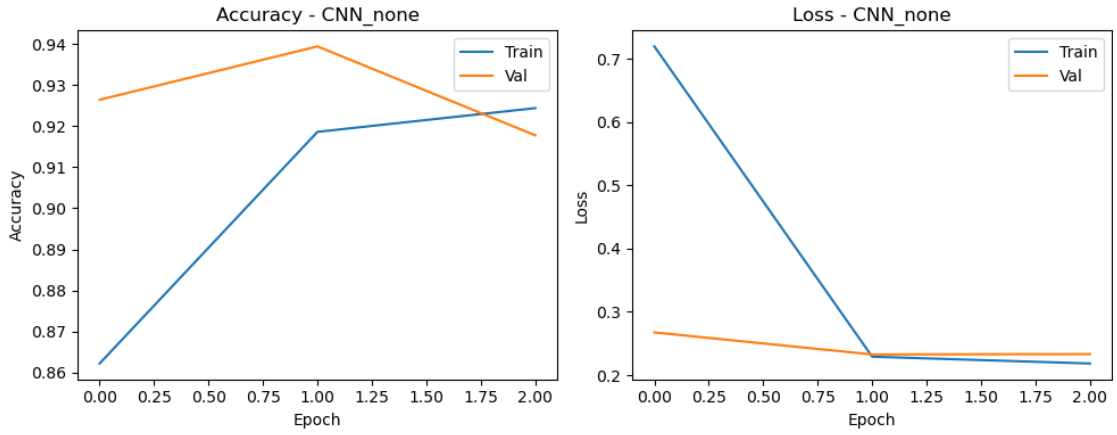
Model Performance Summary:

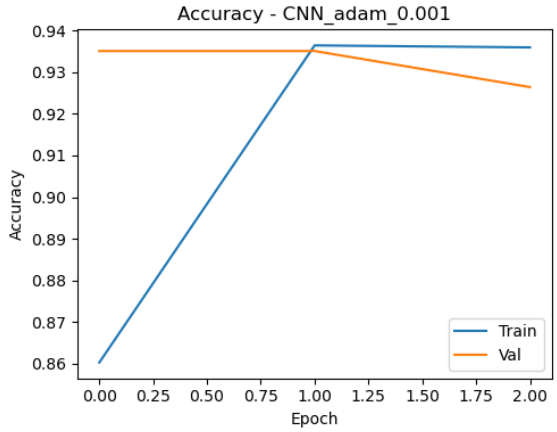
	Model	Optimizer	Learning Rate	Loss	Accuracy
0	VGG16	adam	0.001	0.169650	0.952381
1	LeNet	adam	0.001	0.181951	0.935065
2	CNN	rmsprop	0.001	0.204260	0.930736
3	CNN	adam	0.001	0.208716	0.926407
4	VGG16	None	None	0.197522	0.922078
5	VGG16	sgd	0.001	0.252520	0.922078
6	CNN	None	None	0.233300	0.917749
7	LeNet	rmsprop	0.001	0.214176	0.917749
8	VGG16	adam	0.01	0.230412	0.900433
9	LeNet	None	None	0.244443	0.896104
10	DNN	sgd	0.001	0.261744	0.891775
11	LeNet	adam	0.01	0.219651	0.887446
12	CNN	rmsprop	0.01	0.293018	0.887446
13	CNN	sgd	0.01	0.346322	0.883117
14	DNN	adam	0.001	0.251897	0.883117
15	DNN	sgd	0.01	0.264009	0.878788

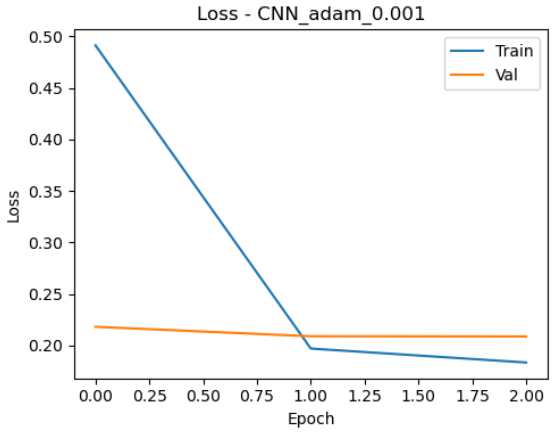
16	LeNet	sgd	0.01	0.257095	0.878788
17	VGG16	rmsprop	0.01	0.258711	0.865801
18	VGG16	rmsprop	0.001	0.302418	0.861472
19	DNN	adam	0.01	0.322788	0.861472
20	CNN	sgd	0.001	0.359139	0.857143
21	DNN	None	None	0.349425	0.857143
22	DNN	rmsprop	0.001	0.286052	0.852814
23	VGG16	sgd	0.01	0.289094	0.852814
24	LeNet	sgd	0.001	0.611087	0.839827
25	ResNet	adam	0.001	0.588431	0.606061
26	DNN	rmsprop	0.01	0.607482	0.606061
27	ResNet	rmsprop	0.001	0.642516	0.567100
28	ResNet	None	None	0.664583	0.549784
29	ResNet	adam	0.01	0.693058	0.510823
30	LeNet	rmsprop	0.01	13.165438	0.510823
31	CNN	adam	0.01	0.694376	0.489177
32	ResNet	sgd	0.001	0.692462	0.489177
33	ResNet	sgd	0.01	0.725449	0.489177
34	ResNet	rmsprop	0.01	0.693177	0.489177

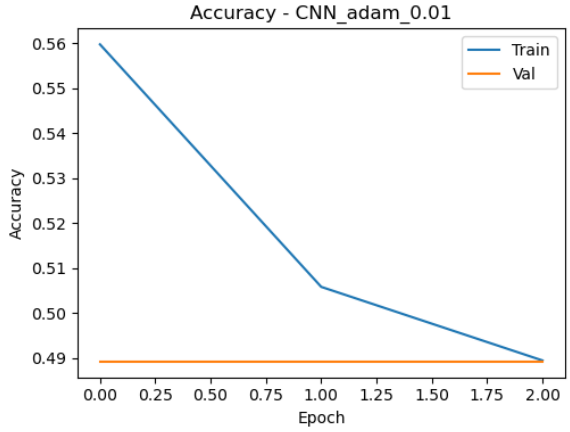
[19]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# --------------------------------
# 1. Plot Accuracy & Loss per Model
# --------------------------------
for key, history in history_logs.items():
    plt.figure(figsize=(10, 4))

    # Accuracy
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train')
    plt.plot(history.history['val_accuracy'], label='Val')
    plt.title(f'Accuracy - {key}')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    # Loss
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Train')
    plt.plot(history.history['val_loss'], label='Val')
    plt.title(f'Loss - {key}')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
```

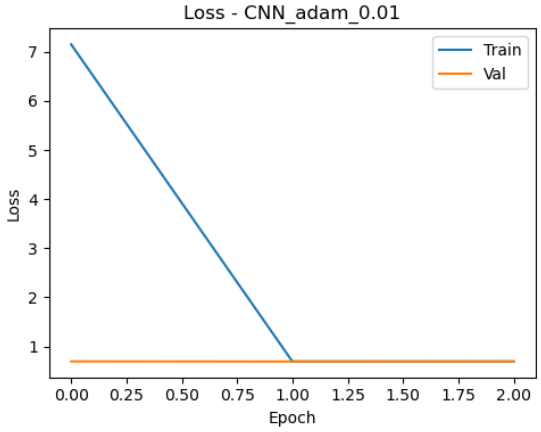
```
    plt.tight_layout()
    plt.show()

# --------------------------------
# 2. Merged Accuracy Graph (only)
# --------------------------------
plt.figure(figsize=(14, 7))
for key, history in history_logs.items():
    plt.plot(history.history['val_accuracy'], label=key)
plt.title('  Validation Accuracy Comparison Across Models')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right', fontsize='small')
plt.grid(True)
plt.tight_layout()
plt.show()

# --------------------------------
# 3. Summary Table
# --------------------------------
summary_df = pd.DataFrame(results, columns=['Model', 'Optimizer', 'Learning␣
 ↪Rate', 'Loss', 'Accuracy'])
summary_df = summary_df.sort_values(by='Accuracy', ascending=False).
 ↪reset_index(drop=True)

print("\n  Model Performance Summary:")
display(summary_df)

# Optional: Save report
# summary_df.to_csv("model_accuracy_summary.csv", index=False)
```

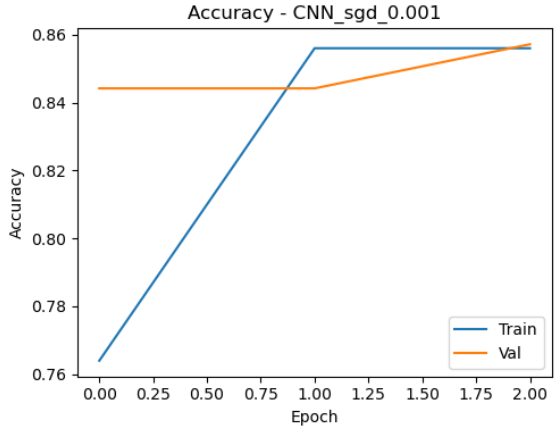


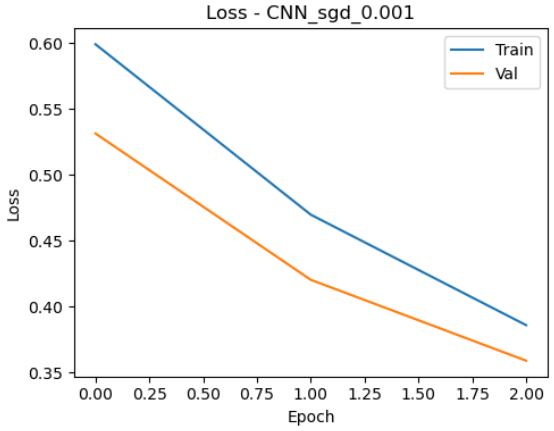

Accuracy - CNN_adam_0.001
Accuracy
Epoch
Train
Val

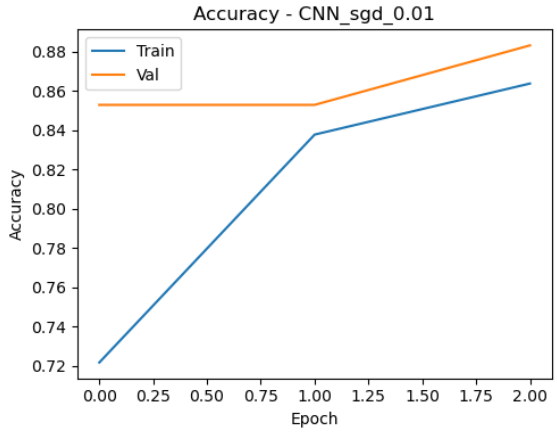

Loss - CNN_adam_0.001
Loss
Epoch
Train
Val

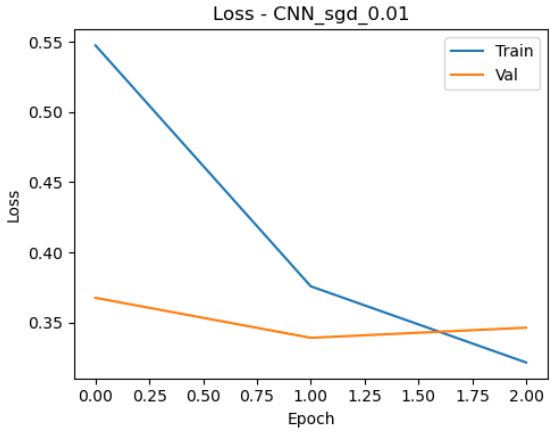

Accuracy - CNN_adam_0.01
Accuracy
Epoch
Train
Val

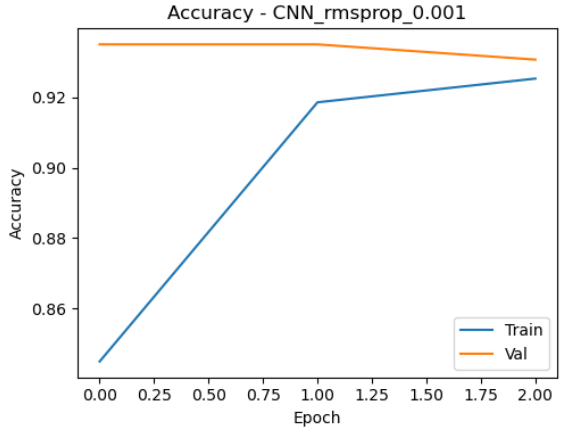

Loss - CNN_adam_0.01
Loss
Epoch
Train
Val

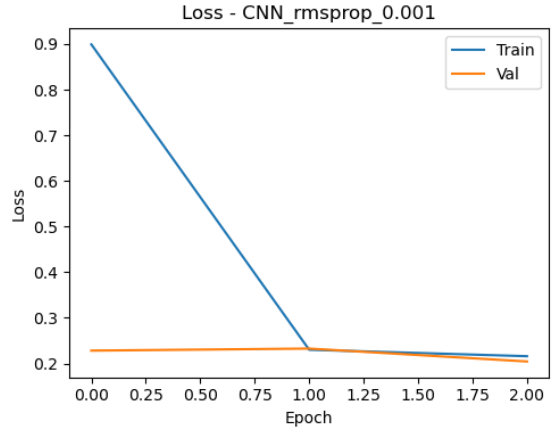

Accuracy - CNN_sgd_0.001
Accuracy
Epoch
Train
Val

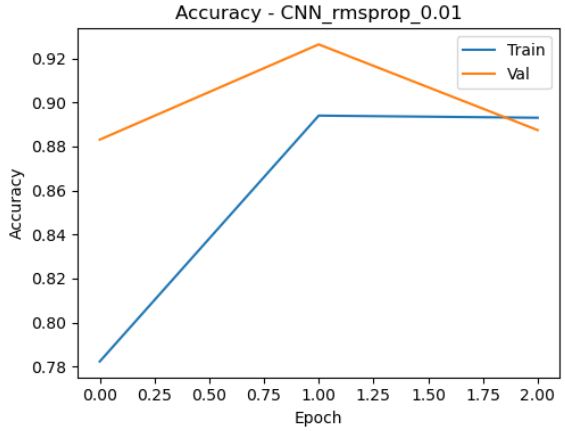

Loss - CNN_sgd_0.001
Loss
Epoch
Train
Val

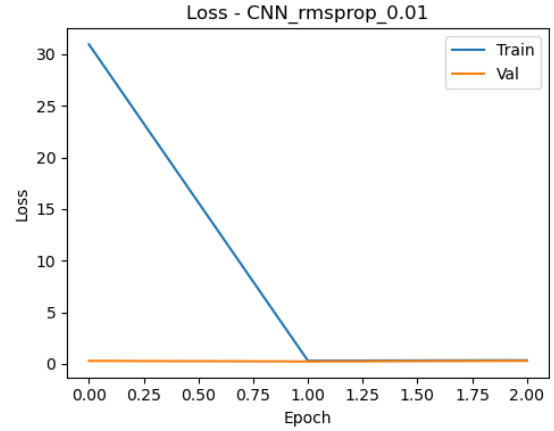

Accuracy - CNN_sgd_0.01
Train
Val
Accuracy
Epoch

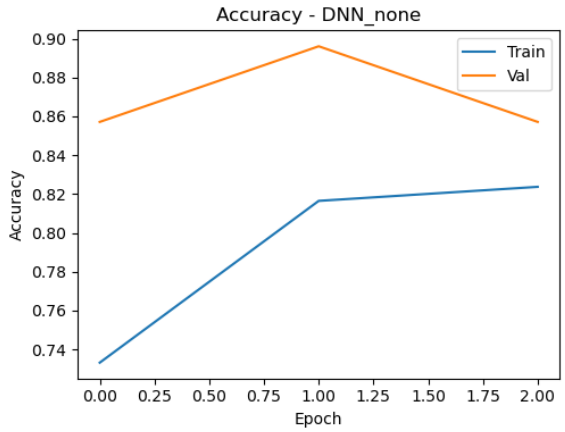

Loss - CNN_sgd_0.01
Train
Val
Loss
Epoch

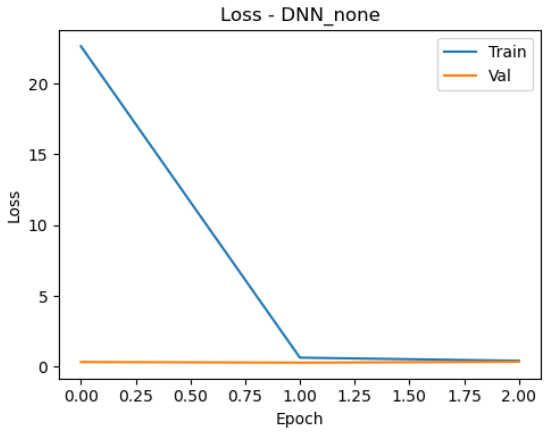

Accuracy - CNN_rmsprop_0.001
Train
Val
Accuracy
Epoch

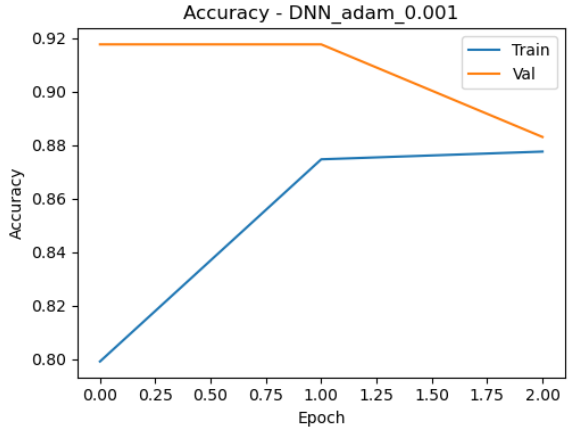

Loss - CNN_rmsprop_0.001
Train
Val
Loss
Epoch

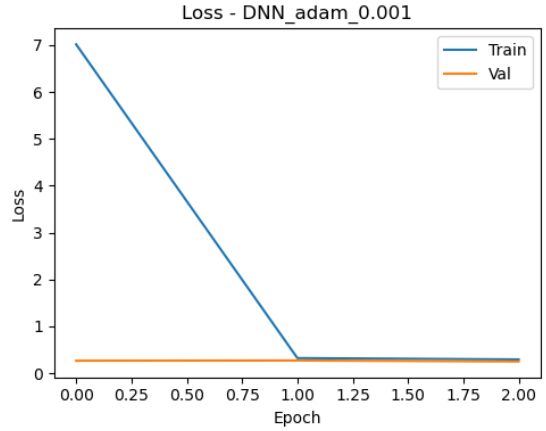

Accuracy - CNN_rmsprop_0.01
Train
Val
Accuracy
Epoch

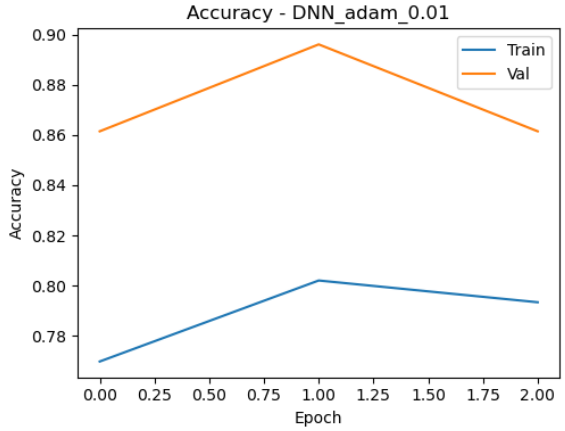

Loss - CNN_rmsprop_0.01
Train
Val
Loss
Epoch

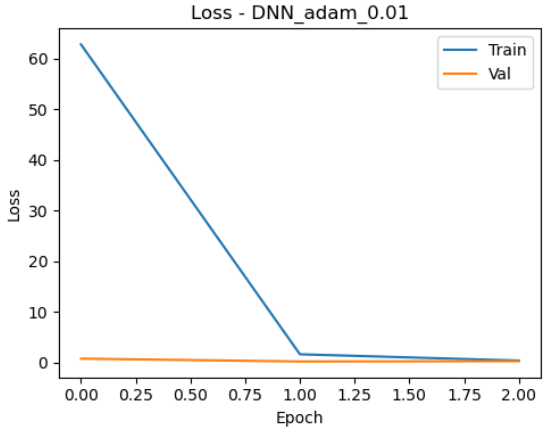

Accuracy - DNN_none
Train
Val
Accuracy
Epoch
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0.74
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

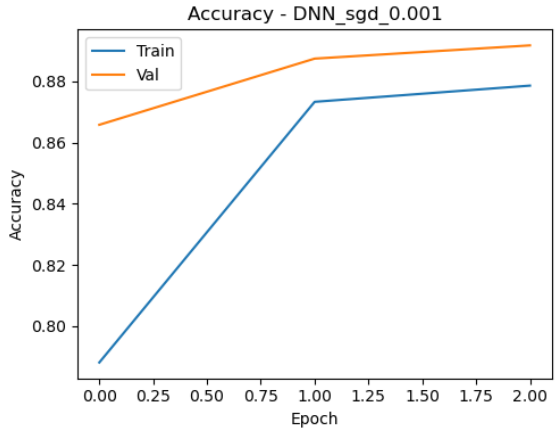

Loss - DNN_none
Train
Val
Loss
Epoch
20
15
10
5
0
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

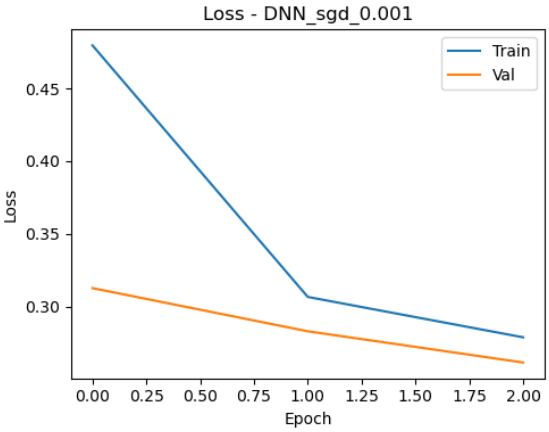

Accuracy - DNN_adam_0.001
Train
Val
Accuracy
Epoch
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

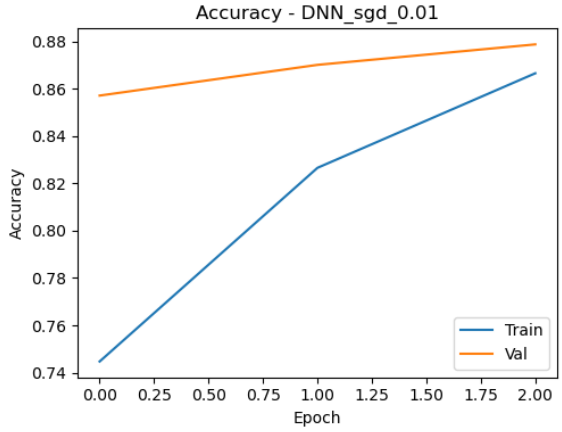

Loss - DNN_adam_0.001
Train
Val
Loss
Epoch
7
6
5
4
3
2
1
0
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

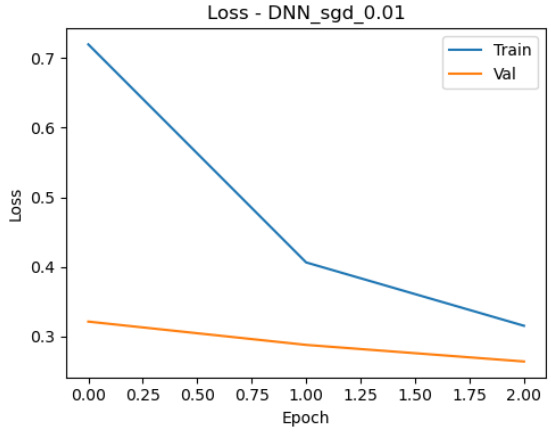

Accuracy - DNN_adam_0.01
Train
Val
Accuracy
Epoch
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

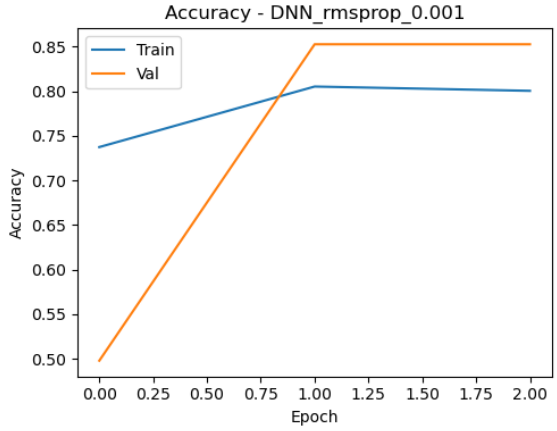

Loss - DNN_adam_0.01
Train
Val
Loss
Epoch
60
50
40
30
20
10
0
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

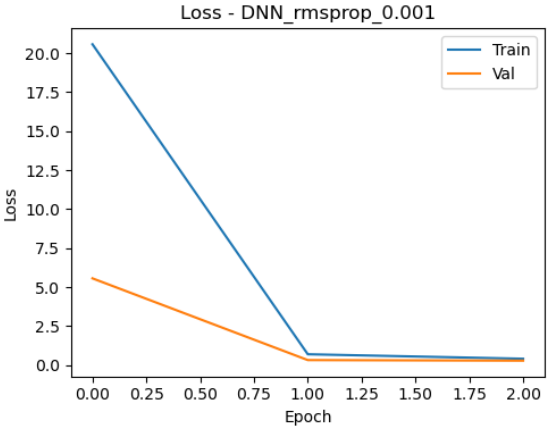

Accuracy - DNN_sgd_0.001
Train
Val
Accuracy
Epoch

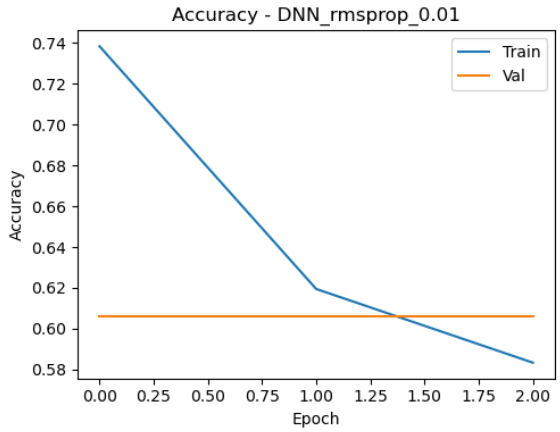

Loss - DNN_sgd_0.001
Train
Val
Loss
Epoch

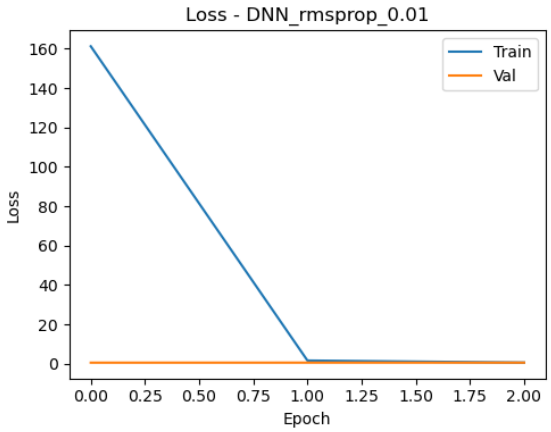

Accuracy - DNN_sgd_0.01
Train
Val
Accuracy
Epoch

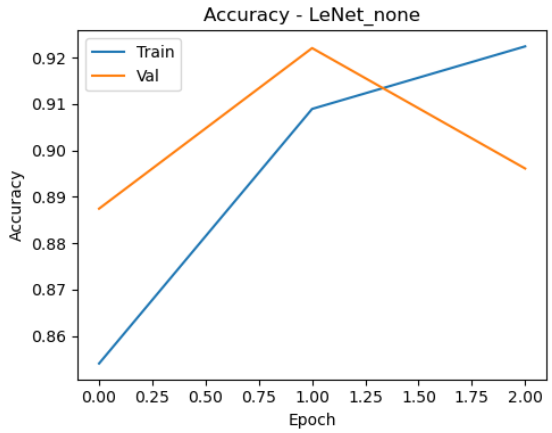

Loss - DNN_sgd_0.01
Train
Val
Loss
Epoch

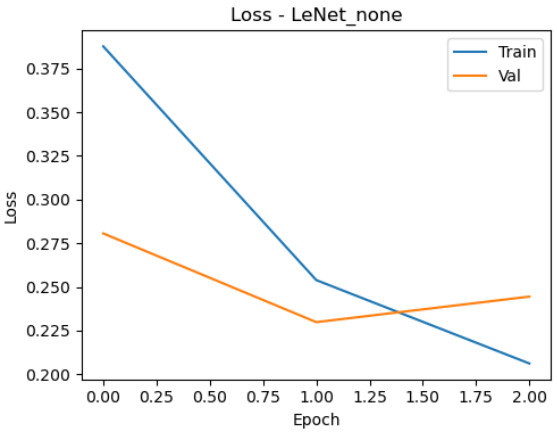

Accuracy - DNN_rmsprop_0.001
Train
Val
Accuracy
Epoch

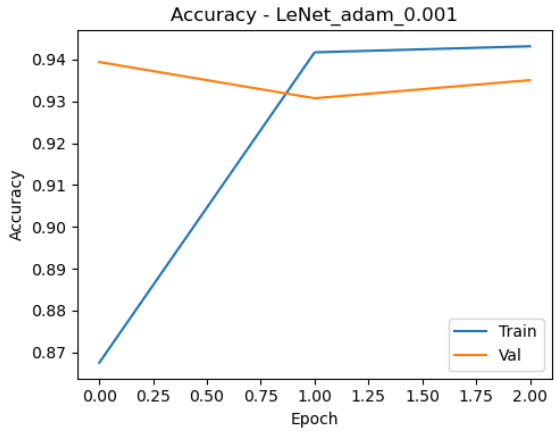

Loss - DNN_rmsprop_0.001
Train
Val
Loss
Epoch

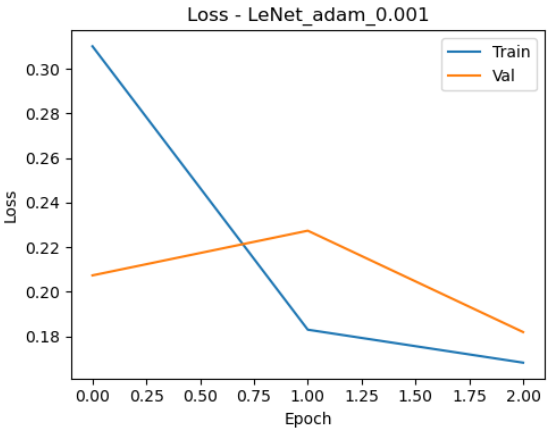

Accuracy - DNN_rmsprop_0.01
Train
Val
Accuracy
Epoch

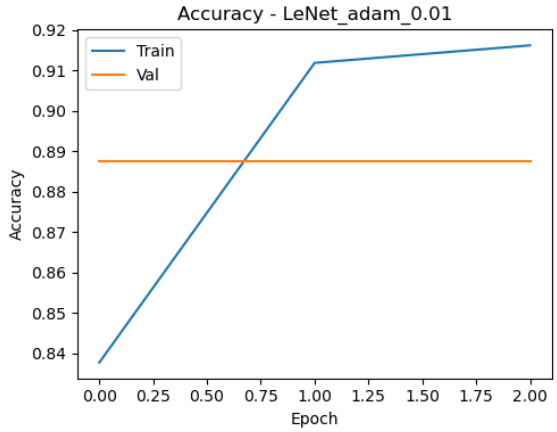

Loss - DNN_rmsprop_0.01
Train
Val
Loss
Epoch

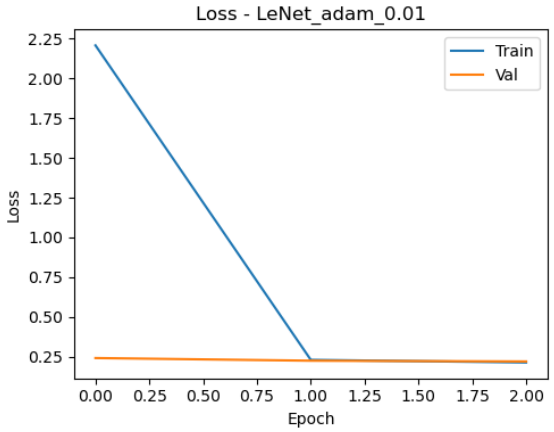

Accuracy - LeNet_none
Train
Val
Accuracy
Epoch

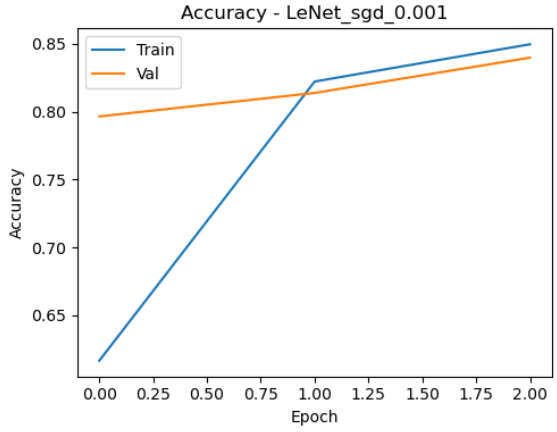

Loss - LeNet_none
Train
Val
Loss
Epoch

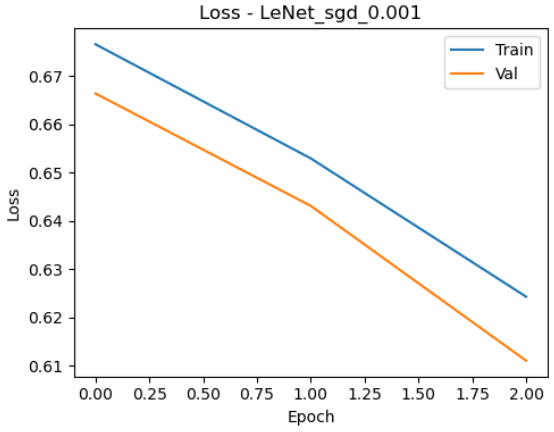

Accuracy - LeNet_adam_0.001
Train
Val
Accuracy
Epoch

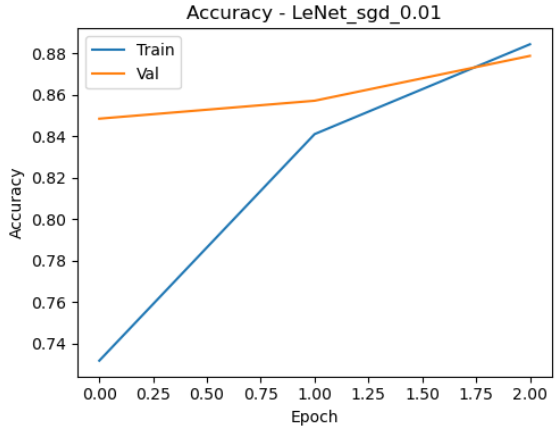

Loss - LeNet_adam_0.001
Train
Val
Loss
Epoch

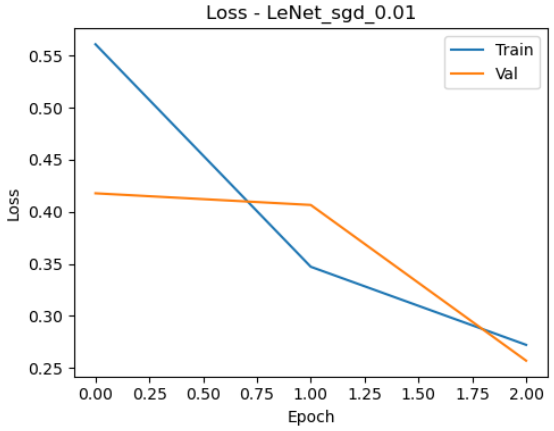

Accuracy - LeNet_adam_0.01
Train
Val
Accuracy
Epoch

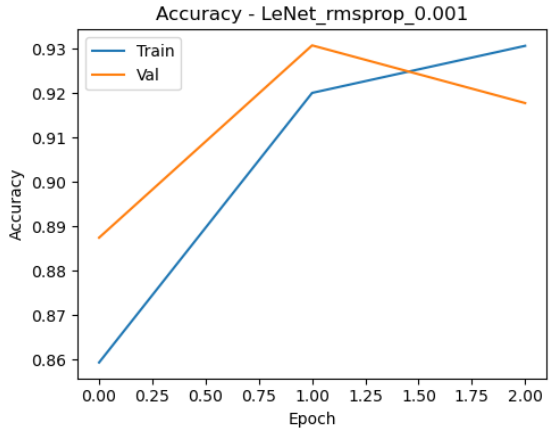

Loss - LeNet_adam_0.01
Train
Val
Loss
Epoch

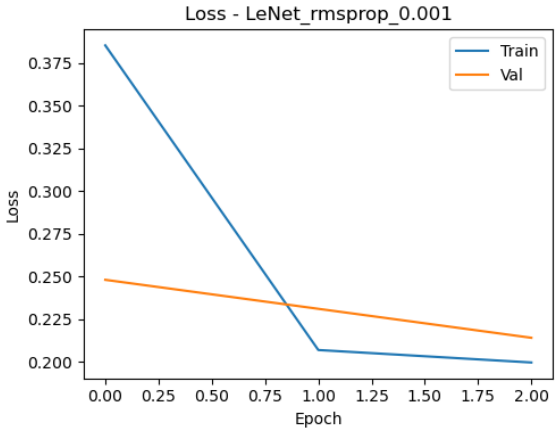

Accuracy - LeNet_sgd_0.001
Train
Val
Accuracy
Epoch

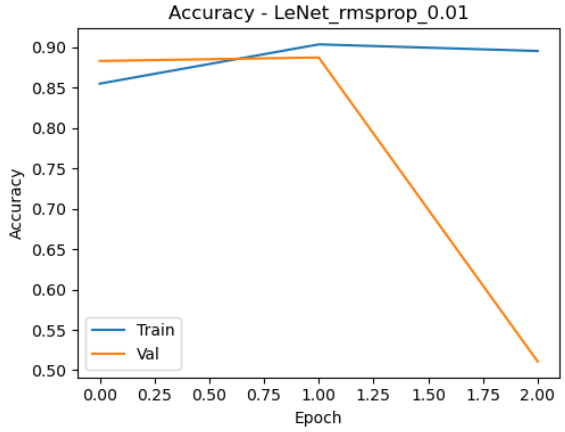

Loss - LeNet_sgd_0.001
Train
Val
Loss
Epoch

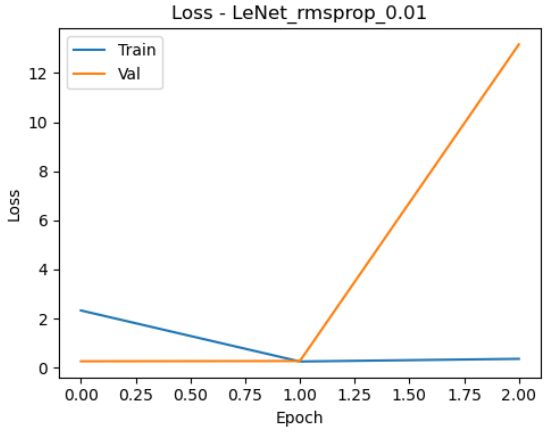

Accuracy - LeNet_sgd_0.01
Train
Val
Accuracy
Epoch

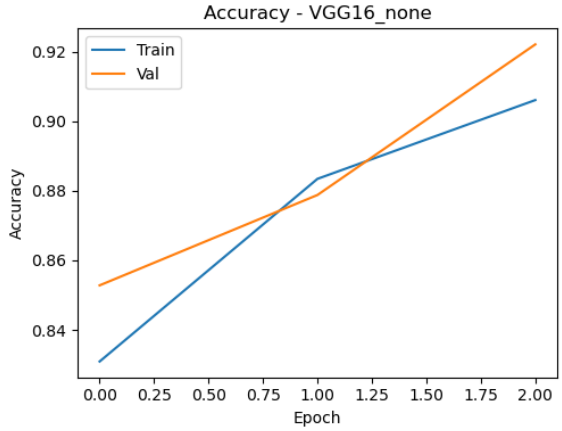

Loss - LeNet_sgd_0.01
Train
Val
Loss
Epoch

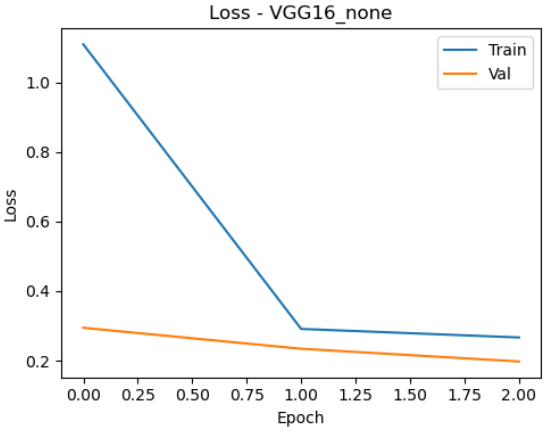

Accuracy - LeNet_rmsprop_0.001
Train
Val
Accuracy
Epoch

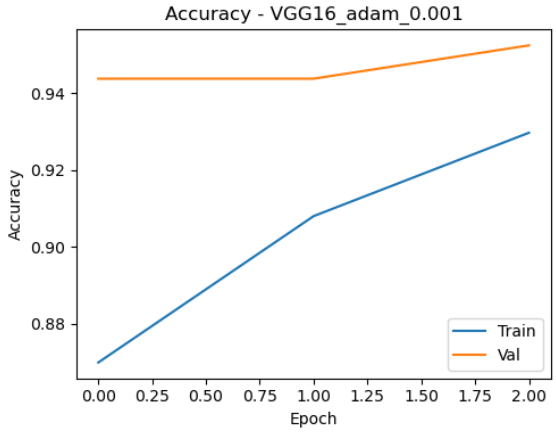

Loss - LeNet_rmsprop_0.001
Train
Val
Loss
Epoch

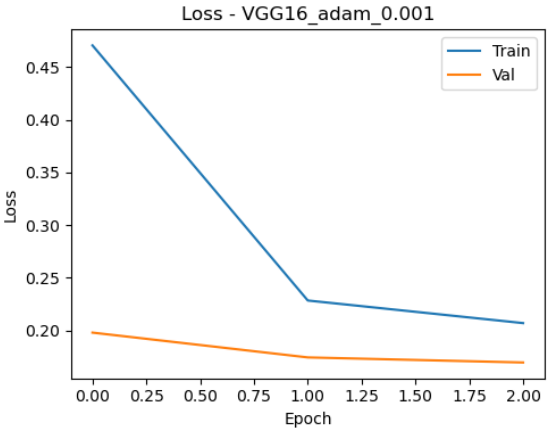

Accuracy - LeNet_rmsprop_0.01
Train
Val
Accuracy
Epoch

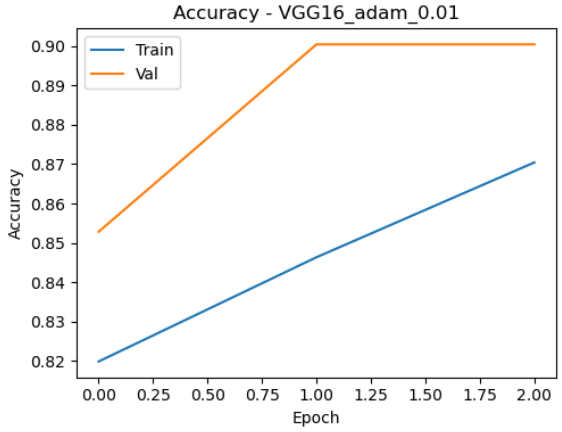

Loss - LeNet_rmsprop_0.01
Train
Val
Loss
Epoch

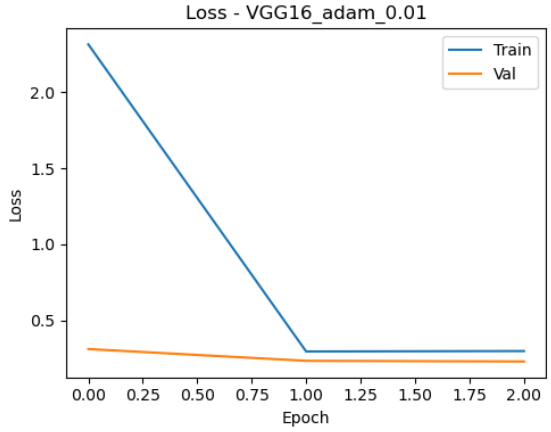

Accuracy - VGG16_none
Train
Val
Accuracy
Epoch

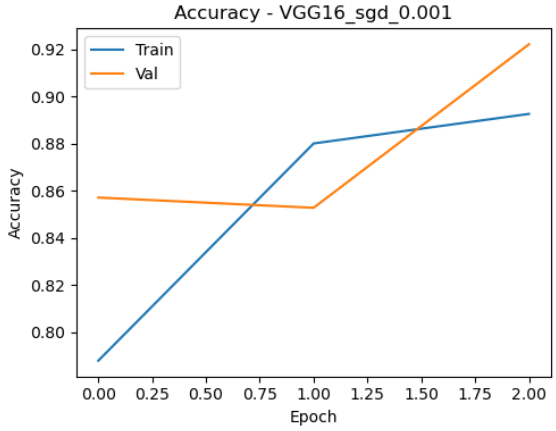

Loss - VGG16_none
Train
Val
Loss
Epoch

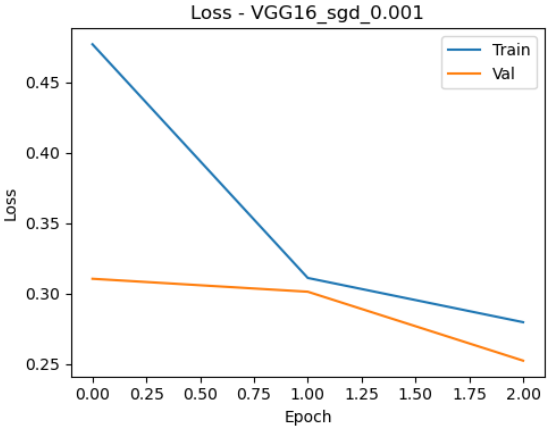

Accuracy - VGG16_adam_0.001
Accuracy
Epoch
Train
Val

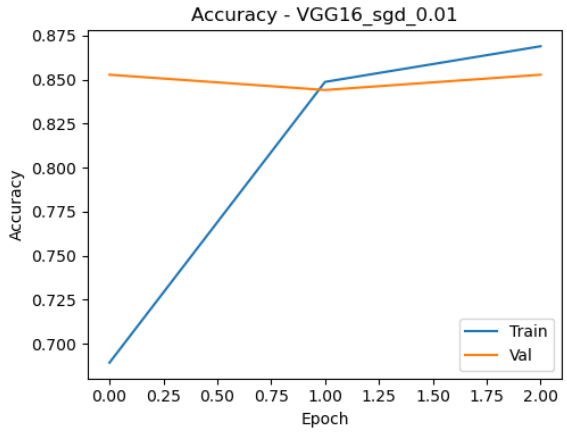

Loss - VGG16_adam_0.001
Loss
Epoch
Train
Val

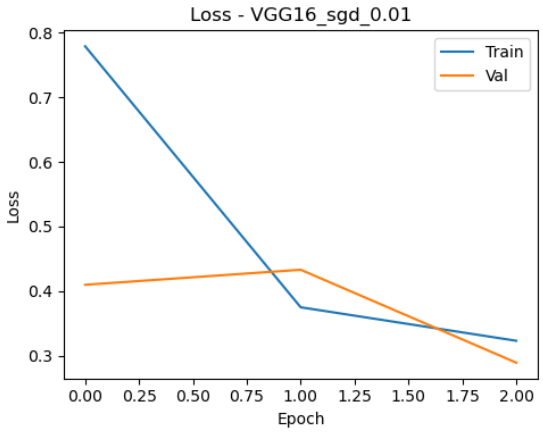

Accuracy - VGG16_adam_0.01
Accuracy
Epoch
Train
Val

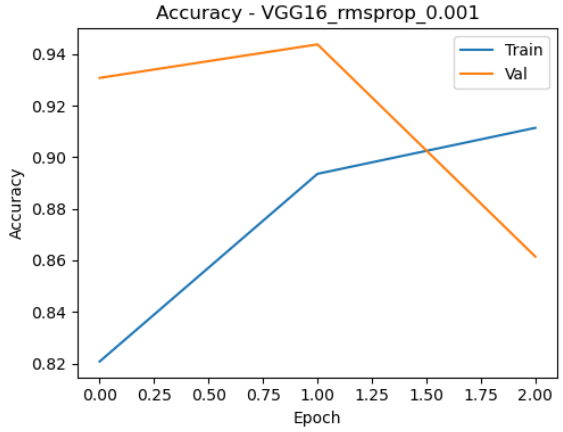

Loss - VGG16_adam_0.01
Loss
Epoch
Train
Val

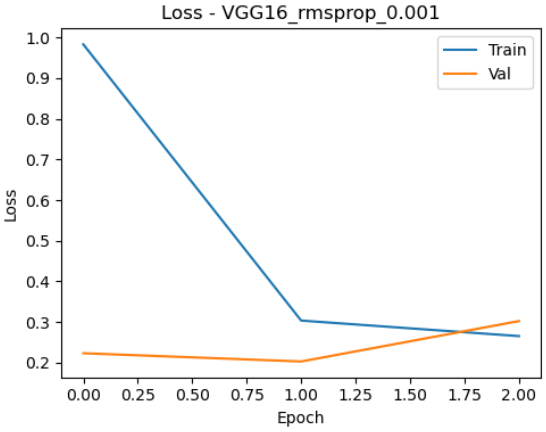

Accuracy - VGG16_sgd_0.001
Accuracy
Epoch
Train
Val

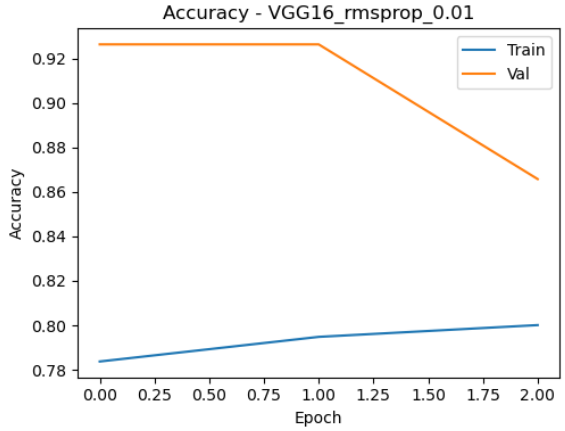

Loss - VGG16_sgd_0.001
Loss
Epoch
Train
Val

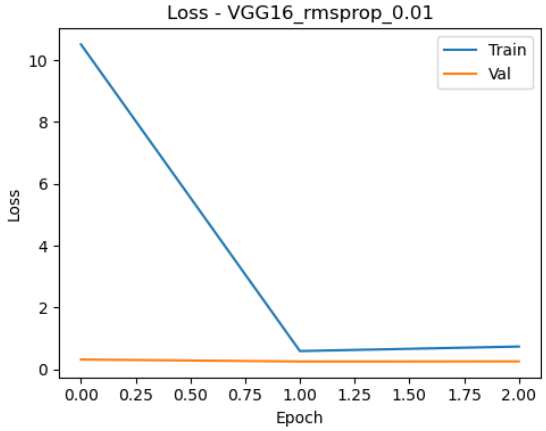

Accuracy - VGG16_sgd_0.01
Accuracy
Epoch
Train
Val

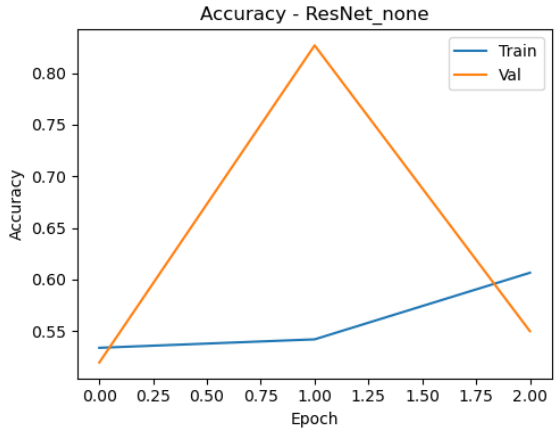

Loss - VGG16_sgd_0.01
Loss
Epoch
Train
Val

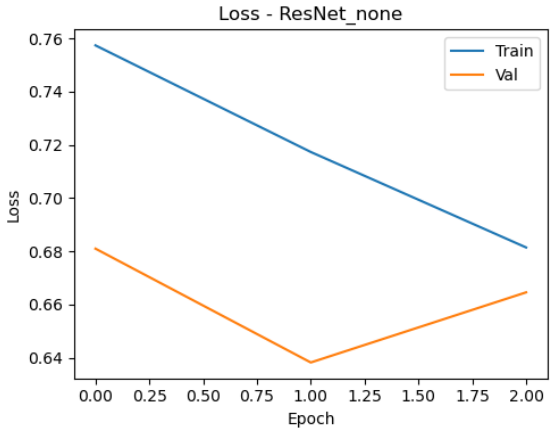

Accuracy - VGG16_rmsprop_0.001
Accuracy
Epoch
Train
Val

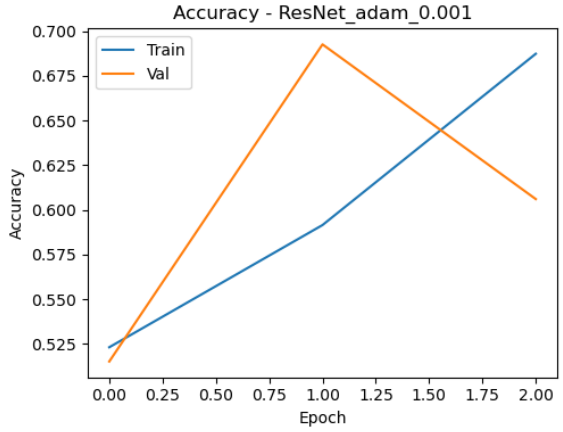

Loss - VGG16_rmsprop_0.001
Loss
Epoch
Train
Val

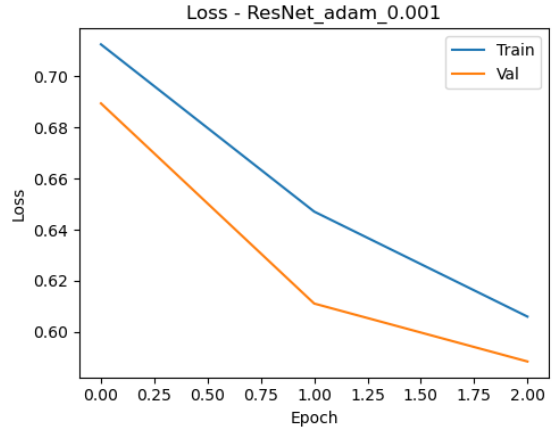

Accuracy - VGG16_rmsprop_0.01
Accuracy
Epoch
Train
Val

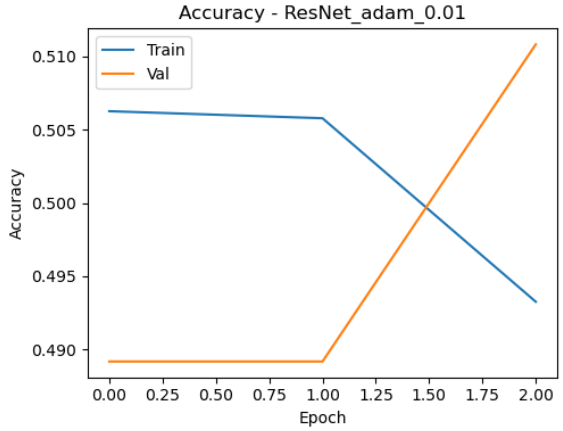

Loss - VGG16_rmsprop_0.01
Loss
Epoch
Train
Val

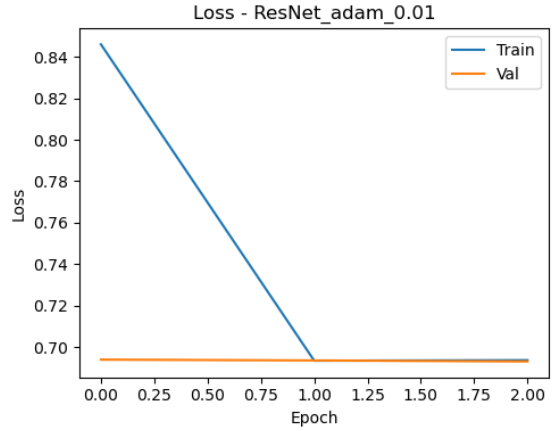

Accuracy - ResNet_none
Train
Val
Accuracy
Epoch

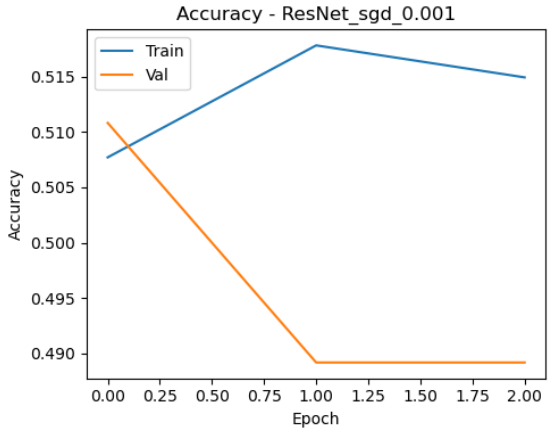

Loss - ResNet_none
Train
Val
Loss
Epoch

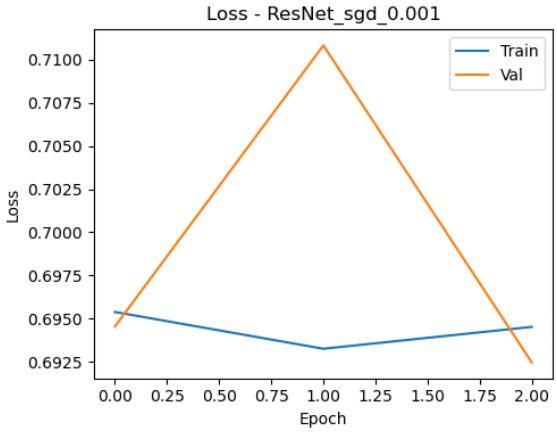

Accuracy - ResNet_adam_0.001
Train
Val
Accuracy
Epoch

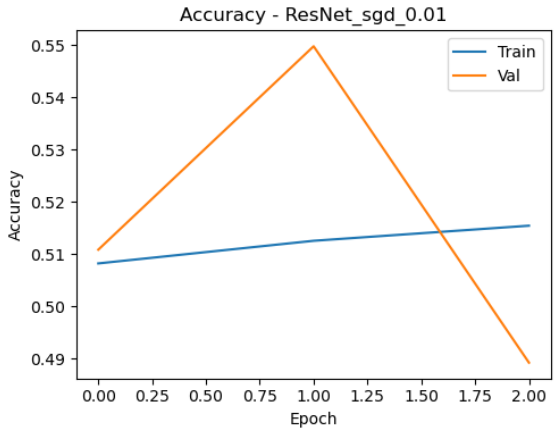

Loss - ResNet_adam_0.001
Train
Val
Loss
Epoch

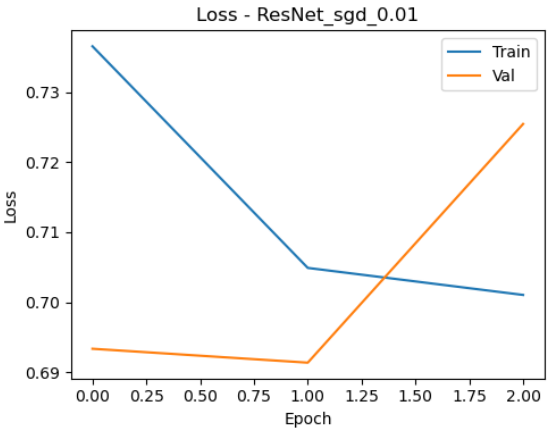

Accuracy - ResNet_adam_0.01
Train
Val
Accuracy
Epoch

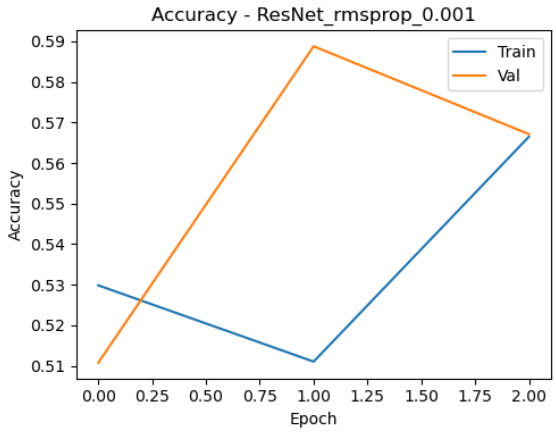

Loss - ResNet_adam_0.01
Train
Val
Loss
Epoch

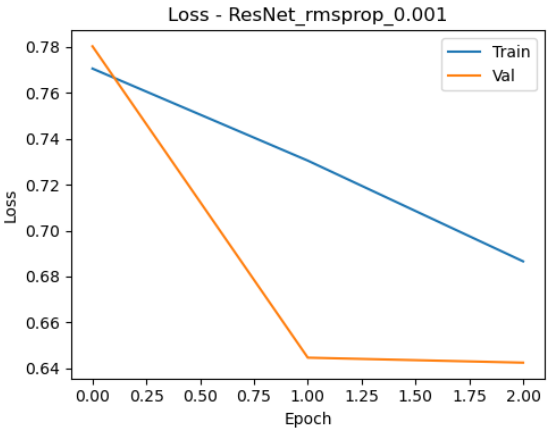

Accuracy - ResNet_sgd_0.001
Train
Val
Accuracy
Epoch

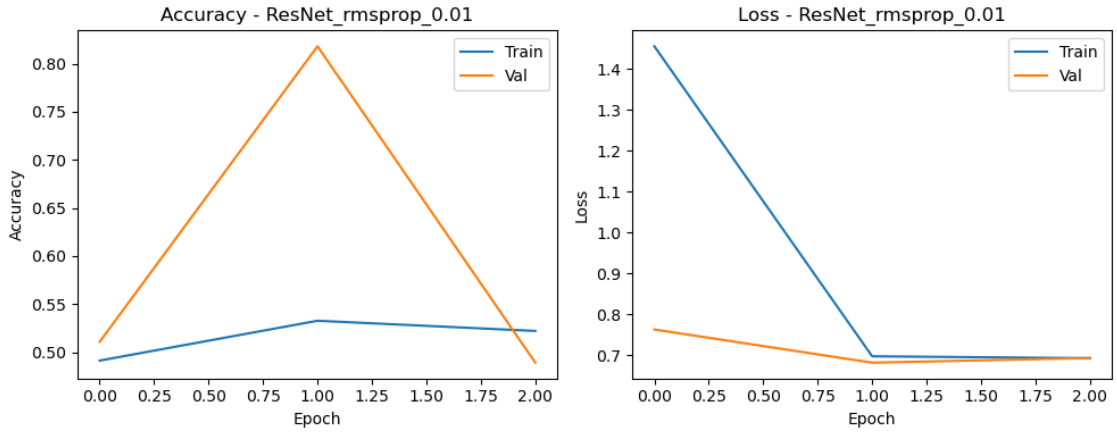

Loss - ResNet_sgd_0.001
Train
Val
Loss
Epoch


Accuracy - ResNet_sgd_0.01
Train
Val
Accuracy
Epoch

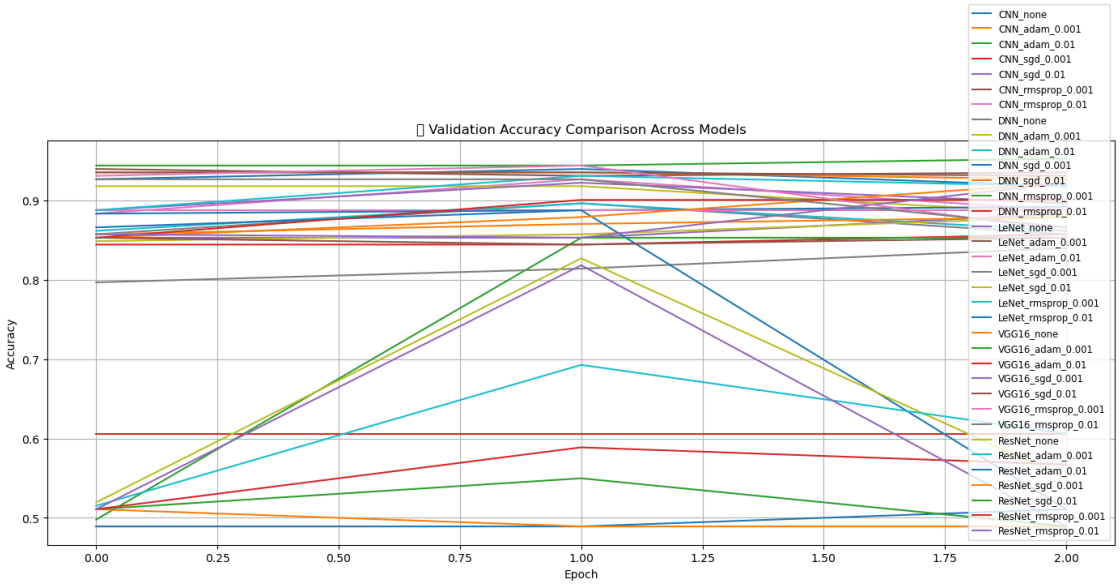

Loss - ResNet_sgd_0.01
Train
Val
Loss
Epoch


Accuracy - ResNet_rmsprop_0.001
Train
Val
Accuracy
Epoch


Loss - ResNet_rmsprop_0.001
Train
Val
Loss
Epoch




```
C:\Users\hp\AppData\Local\Temp\ipykernel_5500\2393719558.py:43: UserWarning:
Glyph 128200 (\N{CHART WITH UPWARDS TREND}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
C:\Users\hp\.conda\envs\dnn\lib\site-packages\IPython\core\pylabtools.py:170:
UserWarning: Glyph 128200 (\N{CHART WITH UPWARDS TREND}) missing from font(s)
DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
  Model Performance Summary:

      Model Optimizer Learning Rate       Loss  Accuracy
0     VGG16      adam          0.001   0.169650  0.952381
```

1	LeNet	adam	0.001	0.181951	0.935065
2	CNN	rmsprop	0.001	0.204260	0.930736
3	CNN	adam	0.001	0.208716	0.926407
4	VGG16	None	None	0.197522	0.922078
5	VGG16	sgd	0.001	0.252520	0.922078
6	CNN	None	None	0.233300	0.917749
7	LeNet	rmsprop	0.001	0.214176	0.917749
8	VGG16	adam	0.01	0.230412	0.900433
9	LeNet	None	None	0.244443	0.896104
10	DNN	sgd	0.001	0.261744	0.891775
11	LeNet	adam	0.01	0.219651	0.887446
12	CNN	rmsprop	0.01	0.293018	0.887446
13	CNN	sgd	0.01	0.346322	0.883117
14	DNN	adam	0.001	0.251897	0.883117
15	DNN	sgd	0.01	0.264009	0.878788
16	LeNet	sgd	0.01	0.257095	0.878788
17	VGG16	rmsprop	0.01	0.258711	0.865801
18	VGG16	rmsprop	0.001	0.302418	0.861472
19	DNN	adam	0.01	0.322788	0.861472
20	CNN	sgd	0.001	0.359139	0.857143
21	DNN	None	None	0.349425	0.857143
22	DNN	rmsprop	0.001	0.286052	0.852814
23	VGG16	sgd	0.01	0.289094	0.852814
24	LeNet	sgd	0.001	0.611087	0.839827
25	ResNet	adam	0.001	0.588431	0.606061
26	DNN	rmsprop	0.01	0.607482	0.606061
27	ResNet	rmsprop	0.001	0.642516	0.567100
28	ResNet	None	None	0.664583	0.549784
29	ResNet	adam	0.01	0.693058	0.510823
30	LeNet	rmsprop	0.01	13.165438	0.510823
31	CNN	adam	0.01	0.694376	0.489177
32	ResNet	sgd	0.001	0.692462	0.489177
33	ResNet	sgd	0.01	0.725449	0.489177
34	ResNet	rmsprop	0.01	0.693177	0.489177

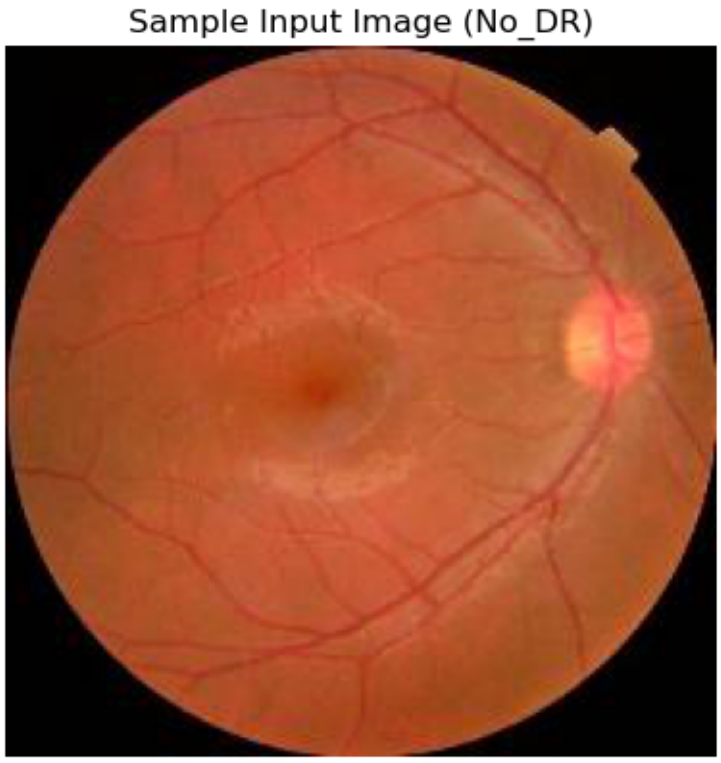
[27]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

# --------------------------------
# 1. Load a sample image
# --------------------------------
img_path = r"C:\Users\hp\Desktop\MTECH_ SEM 2\DNN\Diagnosis of Diabetic␣
 ↪Retinopathy\test\DR\362c4a96cebb_png.rf.192f091fc13737d8c432a2b9c16bd035.jpg"

img = image.load_img(img_path, target_size=(224, 224))  # Resize to match model␣
 ↪input
```

```
# Show image
plt.imshow(img)
plt.axis('off')
plt.title("Sample Input Image")
plt.show()

# --------------------------------
# 2. Preprocess the image
# --------------------------------
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0) / 255.0  # Normalize if needed

# --------------------------------
# 3. Predict using a trained model
# --------------------------------
model_name = 'CNN_adam_0.001'  # use any available trained model
model = trained_models[model_name]

pred = model.predict(img_array)
predicted_class = np.argmax(pred, axis=1)[0]

# --------------------------------
# 4. Display Prediction
# --------------------------------
class_labels = list(test_data.class_indices.keys())  # example: ['dr', 'no_dr']
predicted_label = class_labels[predicted_class]

print(f"  Predicted Label: {predicted_label}")
```


```
1/1             0s 94ms/step
 Predicted Label: DR
```

[28]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

# --------------------------------
# 1. Load the No_DR image
# --------------------------------
img_path = r"C:\Users\hp\Desktop\MTECH_ SEM 2\DNN\Diagnosis of Diabetic␣
 ↪Retinopathy\valid\No_DR\4d167ca69ea8_png.rf.ba34a6446a9941030c1b8601d0211fb1.
 ↪jpg"

img = image.load_img(img_path, target_size=(224, 224))  # Resize to match model␣
 ↪input

# Show image
plt.imshow(img)
plt.axis('off')
plt.title("Sample Input Image (No_DR)")
plt.show()
```

```
# ------------------------------
# 2. Preprocess the image
# ------------------------------
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0) / 255.0  # Normalize

# ------------------------------
# 3. Loop through all models to predict
# ------------------------------
class_labels = list(test_data.class_indices.keys())  # ['dr', 'no_dr']

# Predict with all models
for model_name, model in trained_models.items():
    pred = model.predict(img_array)
    predicted_class = np.argmax(pred, axis=1)[0]
    predicted_label = class_labels[predicted_class]
    print(f"Model: {model_name} - Predicted Label: {predicted_label}")
```




```
1/1             0s 343ms/step
Model: CNN_none - Predicted Label: No_DR
1/1             0s 107ms/step
Model: CNN_adam_0.001 - Predicted Label: No_DR
```

```
1/1             0s 425ms/step
Model: CNN_adam_0.01 - Predicted Label: DR
1/1             0s 384ms/step
Model: CNN_sgd_0.001 - Predicted Label: No_DR
1/1             0s 319ms/step
Model: CNN_sgd_0.01 - Predicted Label: No_DR
1/1             0s 345ms/step
Model: CNN_rmsprop_0.001 - Predicted Label: No_DR
1/1             0s 342ms/step
Model: CNN_rmsprop_0.01 - Predicted Label: No_DR
1/1             0s 383ms/step
Model: DNN_none - Predicted Label: No_DR
1/1             0s 344ms/step
Model: DNN_adam_0.001 - Predicted Label: No_DR
1/1             0s 343ms/step
Model: DNN_adam_0.01 - Predicted Label: No_DR
1/1             0s 500ms/step
Model: DNN_sgd_0.001 - Predicted Label: No_DR
1/1             0s 466ms/step
Model: DNN_sgd_0.01 - Predicted Label: No_DR
1/1             0s 381ms/step
Model: DNN_rmsprop_0.001 - Predicted Label: No_DR
1/1             0s 320ms/step
Model: DNN_rmsprop_0.01 - Predicted Label: DR
1/1             0s 307ms/step
Model: LeNet_none - Predicted Label: No_DR
1/1             0s 292ms/step
Model: LeNet_adam_0.001 - Predicted Label: No_DR
1/1             0s 284ms/step
Model: LeNet_adam_0.01 - Predicted Label: No_DR
1/1             0s 305ms/step
Model: LeNet_sgd_0.001 - Predicted Label: No_DR
1/1             0s 289ms/step
Model: LeNet_sgd_0.01 - Predicted Label: No_DR
1/1             0s 304ms/step
Model: LeNet_rmsprop_0.001 - Predicted Label: No_DR
1/1             0s 258ms/step
Model: LeNet_rmsprop_0.01 - Predicted Label: No_DR
1/1             1s 1s/step
Model: VGG16_none - Predicted Label: No_DR
1/1             1s 965ms/step
Model: VGG16_adam_0.001 - Predicted Label: No_DR
1/1             1s 905ms/step
Model: VGG16_adam_0.01 - Predicted Label: No_DR
1/1             1s 828ms/step
Model: VGG16_sgd_0.001 - Predicted Label: No_DR
1/1             1s 774ms/step
Model: VGG16_sgd_0.01 - Predicted Label: No_DR
```

```
1/1             1s 810ms/step
Model: VGG16_rmsprop_0.001 - Predicted Label: No_DR
1/1             1s 834ms/step
Model: VGG16_rmsprop_0.01 - Predicted Label: No_DR
1/1             4s 4s/step
Model: ResNet_none - Predicted Label: DR
1/1             4s 4s/step
Model: ResNet_adam_0.001 - Predicted Label: No_DR
1/1             3s 3s/step
Model: ResNet_adam_0.01 - Predicted Label: No_DR
1/1             3s 3s/step
Model: ResNet_sgd_0.001 - Predicted Label: DR
1/1             3s 3s/step
Model: ResNet_sgd_0.01 - Predicted Label: DR
1/1             4s 4s/step
Model: ResNet_rmsprop_0.001 - Predicted Label: DR
1/1             4s 4s/step
Model: ResNet_rmsprop_0.01 - Predicted Label: DR
```

[ ]: