all_models

December 4, 2022

0.1 Imports

[ ]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
import os

import seaborn as sns
import matplotlib.pyplot as plt
import sys

import skimage
from skimage.color import rgb2hsv
from skimage.transform import rescale, resize
from tqdm import tqdm

import sys
import os

from sklearn.metrics import classification_report


import bz2, pickle, _pickle as cPickle

import random

# random.seed(1234)

# module_path = os.path.abspath(os.path.join('..'))
# if module_path not in sys.path:
#     sys.path.append(module_path+"/Modules/Testing")
# import testing_module

SAVE_DIR = "../Pickled Datasets/"

COPIES = 2
N_DIGITS = 3
HEIGHT = 25
```

```
WIDTH  = 25


PLOT_SAVE = "../Plots/loss-curves/"

def compressed_pickle(name: str, data):
    with bz2.BZ2File(os.path.join(SAVE_DIR, "{}.pbz2".format(name)), 'w') as f:
        cPickle.dump(data, f)

def decompress_pickle(file):
    data = bz2.BZ2File(file, 'rb')
    data = cPickle.load(data)
    return data

def plot_history(history):
    acc=history.history['accuracy']
    val_acc=history.history['val_accuracy']
    loss=history.history['loss']
    val_loss=history.history['val_loss']
    epochs=range(len(acc))

    fig, ax = plt.subplots(1, 2, figsize = (12, 6))
    ax[0].plot(epochs, acc, 'r', label = "Training Accuracy")
    ax[0].plot(epochs, val_acc, 'b', label = "Validation Accuracy")
    ax[0].legend()
    ax[0].set_title('Training and Validation Accuracy')
    ax[0].set_xlabel("Epochs")

    ax[1].plot(epochs, loss, 'r', label = "Training Loss")
    ax[1].plot(epochs, val_loss, 'b', label = "Validation Loss")
    ax[1].set_title('Training and Validation Losses')
    ax[1].set_xlabel("Epochs")
    plt.show()

    return (fig, ax)
```

SKImage rescales the image for us! Which means that we don't need to rescale by 255.0 anymore, saving us needlessly spent time and effort. There is another Augmentor library which can be used for data augmentation. We can simply sample the augmented images henceforth!

0.2 Preliminary setup

[ ]:
```
# sys.path.append(os.path.dirname(os.path.join((os.path.pardir), "Modules")))

# origin_dir = os.path.join(os.path.pardir, 'Data')
# new_dir_path = os.path.join(os.path.pardir, 'Data', 'cell_images')
```

```
# #for local systems
# train_csv = os.path.join(origin_dir, 'train.csv')
# test_csv = os.path.join(origin_dir, 'test.csv')
# val_csv = os.path.join(origin_dir, 'val.csv')

# from Modules.labelling import Labelling

# # download = Data_Download(origin_dir)
# # data_dir = download.resize_image(new_dir_path, 44, 44)

# lab = Labelling()
# lab.label('../Data/cell_images/', exclude_mislabeled= True)      # function␣
↪to label the dataset
# train_csv, val_csv, test_csv = lab.train_test_val_split('../Data/', '../Data/
↪cell_images/labels.csv', random_state = 1234)

# train_data = pd.read_csv(train_csv)
# val_data   = pd.read_csv(val_csv)
# test_data  = pd.read_csv(test_csv)
```

0.2.1 Reading images

[ ]:
```
# def read_image(path):
#     '''Function to read images given a path and return an array'''
#     return skimage.io.imread(path)

# i = 14

# print(train_data['Image_Path'][i])
# image = rgb2hsv(skimage.io.imread(train_data['Image_Path'][i]))
# print(np.max(image))
# result = ((image > 0.5)*image)[..., 1]
# plt.imshow(result, 'gray')


# tqdm.pandas()
# train_data['image_arr'] = train_data['Image_Path'].progress_apply(lambda x:␣
↪read_image(x))
# val_data['image_arr']   = val_data['Image_Path'].progress_apply(lambda x:␣
↪read_image(x))
# test_data['image_arr']  = test_data['Image_Path'].progress_apply(lambda x:␣
↪read_image(x))


# x_train, y_train = train_data['image_arr'].to_numpy(),␣
↪train_data['Parasitized'].to_numpy()
```

```
# x_val  , y_val   = val_data['image_arr'].to_numpy()  ,␣
↪val_data['Parasitized'].to_numpy()
# x_test , y_test  = test_data['image_arr'].to_numpy() ,␣
↪test_data['Parasitized'].to_numpy()
```

0.3 Data Augmentation

```
[ ]: # import albumentations as A
# import cv2

# augment = A.augmentations.geometric.transforms.Affine(
#     translate_percent = 0.1,
#     rotate = 60,
#     shear = 30
#     )

# augment = A.ShiftScaleRotate(scale_limit = (-0.2, 0), rotate_limit= 90,␣
↪border_mode=cv2.BORDER_CONSTANT, always_apply= True)

# transform = A.Compose(
#     [
#         A.Resize(HEIGHT, WIDTH, always_apply= True),
#         A.Rotate(90, border_mode=cv2.BORDER_CONSTANT),
#         A.VerticalFlip(p = 0.5),
#         A.HorizontalFlip(p = 0.5),
#         A.augmentations.geometric.Affine(shear = 7.5, mode=cv2.
↪BORDER_CONSTANT),
#         A.GaussNoise(var_limit = (0.01, 0.05))
#     ]
# )

# aug_dataset = []
# aug_labels = []


# for i, lab in tqdm(zip(x_train, y_train)):
#     for _ in range(COPIES):
#         aug_dataset.append(transform(image = i)['image'])
#         aug_labels.append(lab)

# x_train_aug = np.array(aug_dataset)
# y_train_aug = np.array(aug_labels)

# np.unique(y_train_aug, return_counts = True)
```

0.3.1 Resizing

[ ]:
```
# temp = []
# for img in tqdm(x_train):
#     temp.append(resize(img, (HEIGHT, WIDTH)))
# x_train = np.array(temp)


# temp = []
# for img in tqdm(x_val):
#     temp.append(resize(img, (HEIGHT, WIDTH)))
# x_val = np.array(temp)


# temp = []
# for img in tqdm(x_test):
#     temp.append(resize(img, (HEIGHT, WIDTH)))
# x_test = np.array(temp)
```

0.4 Saving Data

[ ]:
```
# compressed_pickle("x_train_aug", x_train_aug)
# compressed_pickle("y_train_aug", y_train_aug)
# compressed_pickle("x_train", x_train)
# compressed_pickle("y_train", y_train)
# compressed_pickle("x_val", x_val)
# compressed_pickle("y_val", y_val)
# compressed_pickle("x_test", x_test)
# compressed_pickle("y_test", y_test)

# n_aug_train = x_train_aug.shape[0]
# n_train     = x_train.shape[0]
# n_val       = x_val.shape[0]
# n_test      = x_test.shape[0]
```

0.5 Loading Data

[ ]:
```
x_train_aug = decompress_pickle(SAVE_DIR + 'x_train_aug.pbz2')
y_train_aug = decompress_pickle(SAVE_DIR + 'y_train_aug.pbz2')
x_train = decompress_pickle(SAVE_DIR + 'x_train.pbz2')
y_train = decompress_pickle(SAVE_DIR + 'y_train.pbz2')
x_val = decompress_pickle(SAVE_DIR + 'x_val.pbz2')
y_val = decompress_pickle(SAVE_DIR + 'y_val.pbz2')
x_test = decompress_pickle(SAVE_DIR + 'x_test.pbz2')
y_test = decompress_pickle(SAVE_DIR + 'y_test.pbz2')

print("augmented: ", x_train_aug.shape, y_train_aug.shape)
```

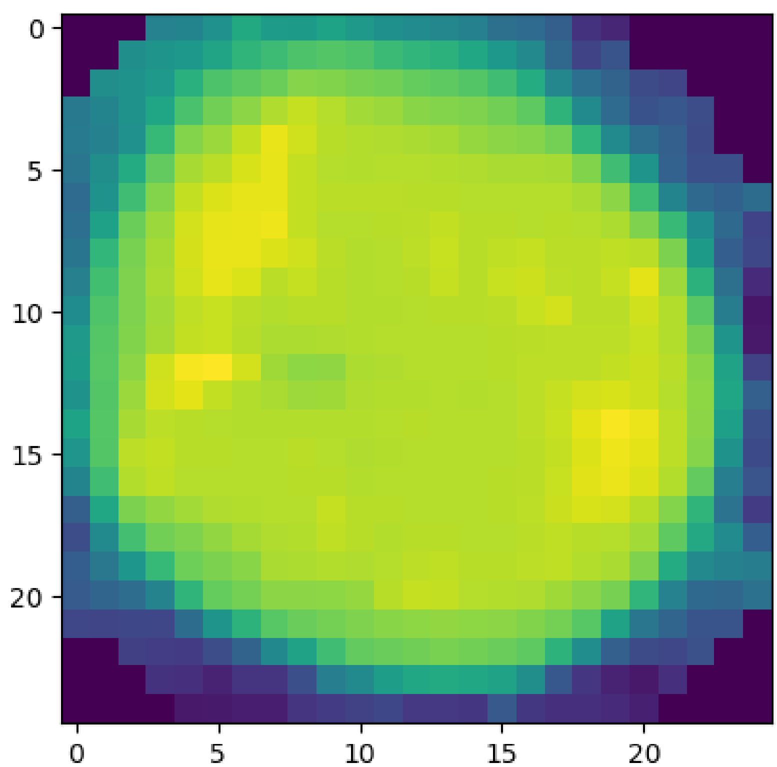
```
print("train: ", x_train.shape, y_train.shape)
print("val: ", x_val.shape, y_val.shape)
print("test: ", x_test.shape, y_test.shape)
```

```
augmented:  (41202, 25, 25, 3) (41202,)
train:  (20601, 25, 25, 3) (20601,)
val:  (2943, 25, 25, 3) (2943,)
test:  (2617, 25, 25, 3) (2617,)
```

[ ]:
```
n_aug_train = x_train_aug.shape[0]
n_train     = x_train.shape[0]
n_val       = x_val.shape[0]
n_test      = x_test.shape[0]
```

[ ]:
```
plt.imshow(x_train[2]), x_train[2].shape
```

[ ]:
```
(<matplotlib.image.AxesImage at 0x1e575f68280>, (25, 25, 1))
```



0.6 RGB Modeling

0.6.1 Unaugmented

Naive Bayes

[ ]:
```python
from sklearn import naive_bayes

nb_cls = naive_bayes.GaussianNB()
nb_cls.fit(x_train.reshape(n_train, -1), y_train)
preds_train = nb_cls.predict(x_train.reshape(n_train, -1))
preds_val   = nb_cls.predict(x_val.reshape(n_val, -1))
preds_test  = nb_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))

# eval = testing_module.ModelEvaluation(y_val,
#     nb_cls.predict(x_val.reshape(n_val, -1)),
#     model_reference_name = 'Naive Bayes',
#     model_type = 'classification',
#     plot_classification_metric = ['roc_auc']) # if classification
# eval.evaluate(evaluate_save= True, plots_show = True)
```

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

         0.0      0.611     0.746     0.672     10260
         1.0      0.677     0.530     0.594     10341

    accuracy                          0.637     20601
   macro avg      0.644     0.638     0.633     20601
weighted avg      0.644     0.637     0.633     20601


Validation Classification Report:
               precision    recall  f1-score   support

         0.0      0.620     0.761     0.683      1466
         1.0      0.693     0.538     0.606      1477

    accuracy                          0.649      2943
   macro avg      0.657     0.649     0.644      2943
weighted avg      0.657     0.649     0.644      2943


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

```
         0.0       0.613     0.766     0.681      1303
         1.0       0.692     0.521     0.595      1314

    accuracy                           0.643      2617
   macro avg       0.653     0.644     0.638      2617
weighted avg       0.653     0.643     0.638      2617
```

Logistic Regression

[ ]:
```
from sklearn.linear_model import LogisticRegression

logreg_cls =  tf.keras.Sequential([tf.keras.layers.Dense(2, activation =␣
 ↪'sigmoid', input_dim = HEIGHT * WIDTH * 3)])
logreg_cls.compile(optimizer = tf.optimizers.Adam(0.001), loss =␣
 ↪'sparse_categorical_crossentropy', metrics = ['accuracy'])
history = logreg_cls.fit(x_train.reshape(n_train, -1),
                         y_train,
                         batch_size = 512,
                         validation_data=[x_val.reshape(n_val, -1), y_val],
                         validation_batch_size=128,
                         epochs = 100,
                         callbacks =  [tf.keras.callbacks.EarlyStopping(monitor␣
 ↪= 'val_loss', patience = 10),
                                       tf.keras.callbacks.ModelCheckpoint(str =␣
 ↪'val_loss', save_best_only = True, save_weights_only = True, filepath=␣
 ↪"LR_No_Aug")],
                         verbose = 1)

logreg_cls.load_weights('LR_No_Aug')
preds_train = np.argmax(logreg_cls.predict(x_train.reshape(n_train, -1)), axis␣
 ↪= 1)
preds_val   = np.argmax(logreg_cls.predict(x_val.reshape(n_val, -1)), axis = 1)
preds_test  = np.argmax(logreg_cls.predict(x_test.reshape(n_test, -1)), axis =␣
 ↪1)

fig, ax = plot_history(history)

fig.savefig(PLOT_SAVE + "logreg_unaug_losscurve.png", facecolor = 'white')

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/100
```

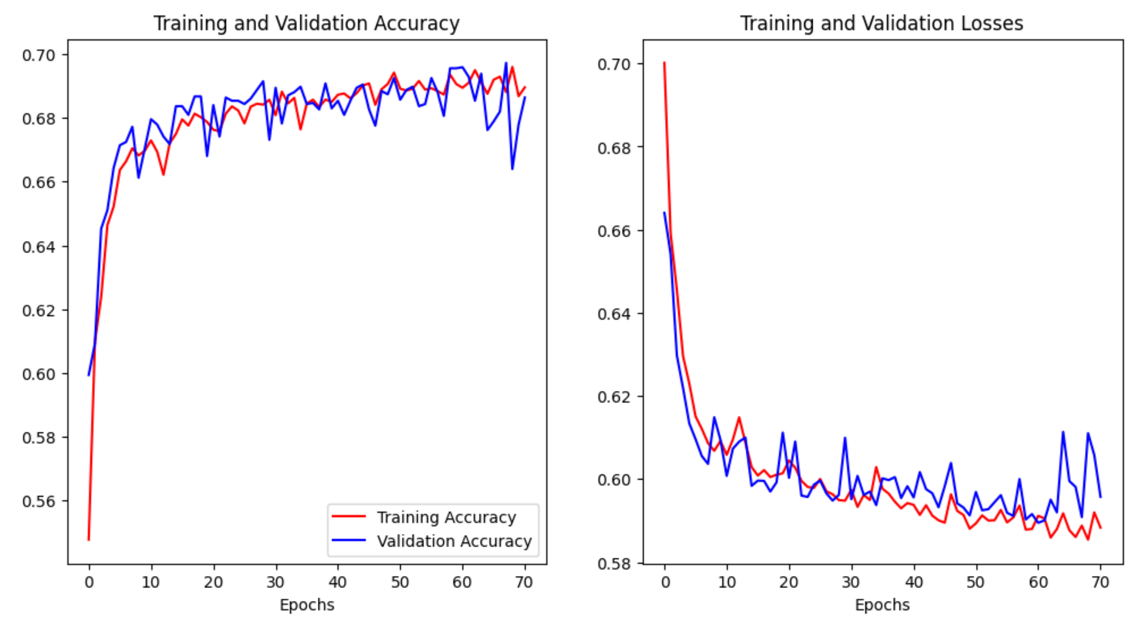
```
41/41 [==============================] - 1s 6ms/step - loss: 0.7001 - accuracy:
0.5477 - val_loss: 0.6641 - val_accuracy: 0.5994
Epoch 2/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6591 - accuracy:
0.6098 - val_loss: 0.6541 - val_accuracy: 0.6092
Epoch 3/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6455 - accuracy:
0.6239 - val_loss: 0.6298 - val_accuracy: 0.6453
Epoch 4/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6295 - accuracy:
0.6464 - val_loss: 0.6219 - val_accuracy: 0.6510
Epoch 5/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6230 - accuracy:
0.6522 - val_loss: 0.6134 - val_accuracy: 0.6643
Epoch 6/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6151 - accuracy:
0.6637 - val_loss: 0.6096 - val_accuracy: 0.6714
Epoch 7/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6121 - accuracy:
0.6664 - val_loss: 0.6055 - val_accuracy: 0.6724
Epoch 8/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6087 - accuracy:
0.6705 - val_loss: 0.6036 - val_accuracy: 0.6772
Epoch 9/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6068 - accuracy:
0.6682 - val_loss: 0.6149 - val_accuracy: 0.6612
Epoch 10/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6092 - accuracy:
0.6697 - val_loss: 0.6094 - val_accuracy: 0.6704
Epoch 11/100
41/41 [==============================] - 0s 4ms/step - loss: 0.6058 - accuracy:
0.6729 - val_loss: 0.6008 - val_accuracy: 0.6796
Epoch 12/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6095 - accuracy:
0.6693 - val_loss: 0.6073 - val_accuracy: 0.6779
Epoch 13/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6148 - accuracy:
0.6622 - val_loss: 0.6090 - val_accuracy: 0.6741
Epoch 14/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6088 - accuracy:
0.6721 - val_loss: 0.6100 - val_accuracy: 0.6718
Epoch 15/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6029 - accuracy:
0.6750 - val_loss: 0.5984 - val_accuracy: 0.6837
Epoch 16/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6008 - accuracy:
0.6795 - val_loss: 0.5996 - val_accuracy: 0.6837
Epoch 17/100
```

```
41/41 [==============================] - 0s 3ms/step - loss: 0.6022 - accuracy:
0.6776 - val_loss: 0.5996 - val_accuracy: 0.6809
Epoch 18/100
41/41 [==============================] - 0s 4ms/step - loss: 0.6005 - accuracy:
0.6813 - val_loss: 0.5970 - val_accuracy: 0.6867
Epoch 19/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6010 - accuracy:
0.6802 - val_loss: 0.5992 - val_accuracy: 0.6867
Epoch 20/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6014 - accuracy:
0.6787 - val_loss: 0.6112 - val_accuracy: 0.6680
Epoch 21/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6045 - accuracy:
0.6763 - val_loss: 0.6003 - val_accuracy: 0.6840
Epoch 22/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6027 - accuracy:
0.6758 - val_loss: 0.6090 - val_accuracy: 0.6741
Epoch 23/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5996 - accuracy:
0.6813 - val_loss: 0.5960 - val_accuracy: 0.6864
Epoch 24/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5981 - accuracy:
0.6836 - val_loss: 0.5957 - val_accuracy: 0.6854
Epoch 25/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5979 - accuracy:
0.6823 - val_loss: 0.5986 - val_accuracy: 0.6854
Epoch 26/100
41/41 [==============================] - 0s 4ms/step - loss: 0.6000 - accuracy:
0.6783 - val_loss: 0.5996 - val_accuracy: 0.6843
Epoch 27/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5971 - accuracy:
0.6835 - val_loss: 0.5966 - val_accuracy: 0.6860
Epoch 28/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5963 - accuracy:
0.6844 - val_loss: 0.5948 - val_accuracy: 0.6888
Epoch 29/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5949 - accuracy:
0.6841 - val_loss: 0.5962 - val_accuracy: 0.6915
Epoch 30/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5948 - accuracy:
0.6857 - val_loss: 0.6099 - val_accuracy: 0.6731
Epoch 31/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5974 - accuracy:
0.6808 - val_loss: 0.5952 - val_accuracy: 0.6894
Epoch 32/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5933 - accuracy:
0.6882 - val_loss: 0.6008 - val_accuracy: 0.6782
Epoch 33/100
```

```
41/41 [==============================] - 0s 3ms/step - loss: 0.5961 - accuracy:
0.6845 - val_loss: 0.5962 - val_accuracy: 0.6871
Epoch 34/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5950 - accuracy:
0.6863 - val_loss: 0.5970 - val_accuracy: 0.6881
Epoch 35/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6029 - accuracy:
0.6764 - val_loss: 0.5937 - val_accuracy: 0.6898
Epoch 36/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5977 - accuracy:
0.6845 - val_loss: 0.6002 - val_accuracy: 0.6843
Epoch 37/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5965 - accuracy:
0.6857 - val_loss: 0.5997 - val_accuracy: 0.6847
Epoch 38/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5945 - accuracy:
0.6834 - val_loss: 0.6005 - val_accuracy: 0.6826
Epoch 39/100
41/41 [==============================] - 0s 4ms/step - loss: 0.5930 - accuracy:
0.6857 - val_loss: 0.5954 - val_accuracy: 0.6908
Epoch 40/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5942 - accuracy:
0.6850 - val_loss: 0.5983 - val_accuracy: 0.6830
Epoch 41/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5938 - accuracy:
0.6872 - val_loss: 0.5956 - val_accuracy: 0.6854
Epoch 42/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5914 - accuracy:
0.6876 - val_loss: 0.6017 - val_accuracy: 0.6809
Epoch 43/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5937 - accuracy:
0.6861 - val_loss: 0.5976 - val_accuracy: 0.6854
Epoch 44/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5912 - accuracy:
0.6877 - val_loss: 0.5966 - val_accuracy: 0.6894
Epoch 45/100
41/41 [==============================] - 0s 4ms/step - loss: 0.5901 - accuracy:
0.6902 - val_loss: 0.5932 - val_accuracy: 0.6905
Epoch 46/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5895 - accuracy:
0.6908 - val_loss: 0.5983 - val_accuracy: 0.6826
Epoch 47/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5963 - accuracy:
0.6841 - val_loss: 0.6039 - val_accuracy: 0.6775
Epoch 48/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5923 - accuracy:
0.6889 - val_loss: 0.5942 - val_accuracy: 0.6884
Epoch 49/100
```

```
41/41 [==============================] - 0s 4ms/step - loss: 0.5914 - accuracy:
0.6906 - val_loss: 0.5931 - val_accuracy: 0.6874
Epoch 50/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5881 - accuracy:
0.6942 - val_loss: 0.5912 - val_accuracy: 0.6925
Epoch 51/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5893 - accuracy:
0.6891 - val_loss: 0.5969 - val_accuracy: 0.6857
Epoch 52/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5912 - accuracy:
0.6886 - val_loss: 0.5925 - val_accuracy: 0.6888
Epoch 53/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5900 - accuracy:
0.6891 - val_loss: 0.5928 - val_accuracy: 0.6898
Epoch 54/100
41/41 [==============================] - 0s 4ms/step - loss: 0.5901 - accuracy:
0.6916 - val_loss: 0.5944 - val_accuracy: 0.6837
Epoch 55/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5926 - accuracy:
0.6889 - val_loss: 0.5961 - val_accuracy: 0.6843
Epoch 56/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5896 - accuracy:
0.6893 - val_loss: 0.5919 - val_accuracy: 0.6925
Epoch 57/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5908 - accuracy:
0.6884 - val_loss: 0.5911 - val_accuracy: 0.6881
Epoch 58/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5936 - accuracy:
0.6873 - val_loss: 0.6000 - val_accuracy: 0.6806
Epoch 59/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5878 - accuracy:
0.6936 - val_loss: 0.5903 - val_accuracy: 0.6955
Epoch 60/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5880 - accuracy:
0.6906 - val_loss: 0.5916 - val_accuracy: 0.6955
Epoch 61/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5912 - accuracy:
0.6894 - val_loss: 0.5895 - val_accuracy: 0.6959
Epoch 62/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5905 - accuracy:
0.6910 - val_loss: 0.5900 - val_accuracy: 0.6928
Epoch 63/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5859 - accuracy:
0.6950 - val_loss: 0.5950 - val_accuracy: 0.6854
Epoch 64/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5880 - accuracy:
0.6911 - val_loss: 0.5920 - val_accuracy: 0.6938
Epoch 65/100
```

```
41/41 [==============================] - 0s 3ms/step - loss: 0.5918 - accuracy:
0.6875 - val_loss: 0.6114 - val_accuracy: 0.6762
Epoch 66/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5877 - accuracy:
0.6920 - val_loss: 0.5995 - val_accuracy: 0.6789
Epoch 67/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5861 - accuracy:
0.6929 - val_loss: 0.5981 - val_accuracy: 0.6820
Epoch 68/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5888 - accuracy:
0.6881 - val_loss: 0.5908 - val_accuracy: 0.6972
Epoch 69/100
41/41 [==============================] - 0s 4ms/step - loss: 0.5854 - accuracy:
0.6960 - val_loss: 0.6110 - val_accuracy: 0.6639
Epoch 70/100
41/41 [==============================] - 0s 4ms/step - loss: 0.5920 - accuracy:
0.6868 - val_loss: 0.6058 - val_accuracy: 0.6779
Epoch 71/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5883 - accuracy:
0.6896 - val_loss: 0.5957 - val_accuracy: 0.6864
644/644 [==============================] - 0s 701us/step
92/92 [==============================] - 0s 709us/step
82/82 [==============================] - 0s 716us/step
```




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

```
         0.0      0.701     0.687     0.694     10260
         1.0      0.695     0.710     0.703     10341

    accuracy                          0.698     20601
   macro avg      0.698     0.698     0.698     20601
weighted avg      0.698     0.698     0.698     20601


Validation Classification Report:
               precision    recall  f1-score   support

         0.0      0.696     0.692     0.694      1466
         1.0      0.696     0.699     0.698      1477

    accuracy                          0.696      2943
   macro avg      0.696     0.696     0.696      2943
weighted avg      0.696     0.696     0.696      2943


Testing Classification Report:
               precision    recall  f1-score   support

         0.0      0.693     0.691     0.692      1303
         1.0      0.695     0.696     0.696      1314

    accuracy                          0.694      2617
   macro avg      0.694     0.694     0.694      2617
weighted avg      0.694     0.694     0.694      2617
```

Decision Trees

[ ]:
```
from sklearn.tree import DecisionTreeClassifier

dt_cls =  DecisionTreeClassifier(criterion = 'gini', splitter = 'best',␣
 ↪max_depth = 5, min_samples_split = 2, )
dt_cls.fit(x_train.reshape(n_train, -1), y_train)
preds_train = dt_cls.predict(x_train.reshape(n_train, -1))
preds_val   = dt_cls.predict(x_val.reshape(n_val, -1))
preds_test  = dt_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
# eval = testing_module.ModelEvaluation(y_val,
#     nb_cls.predict(x_val.reshape(n_val, -1)),
#     model_reference_name = 'Naive Bayes',
#     model_type = 'classification',
#     plot_classification_metric = ['roc_auc']) # if classification
# eval.evaluate(evaluate_save= True, plots_show = True)
```

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

         0.0      0.685     0.774     0.727     10260
         1.0      0.743     0.646     0.691     10341

    accuracy                          0.710     20601
   macro avg      0.714     0.710     0.709     20601
weighted avg      0.714     0.710     0.709     20601


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.676     0.774     0.721      1466
         1.0      0.738     0.632     0.681      1477

    accuracy                          0.702      2943
   macro avg      0.707     0.703     0.701      2943
weighted avg      0.707     0.702     0.701      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.661     0.748     0.702      1303
         1.0      0.713     0.619     0.663      1314

    accuracy                          0.684      2617
   macro avg      0.687     0.684     0.682      2617
weighted avg      0.687     0.684     0.682      2617
```

XGBoost

[ ]:
```
tf.keras.backend.clear_session()

from xgboost import XGBClassifier

xgb_cls = XGBClassifier(max_depth = 5, objective = 'reg:logistic',
                            num_parallel_tree = 20, booster = 'gbtree',
```

```
                            gamma = 0.5, tree_method = 'gpu_hist', subsample =␣
 ↪0.4, reg_lambda = 1)
xgb_cls.fit(x_train.reshape(n_train, -1), y_train)
preds_train = xgb_cls.predict(x_train.reshape(n_train, -1))
preds_val   = xgb_cls.predict(x_val.reshape(n_val, -1))
preds_test  = xgb_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

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

         0.0      0.966     0.983     0.974     10260
         1.0      0.983     0.966     0.974     10341

    accuracy                          0.974     20601
   macro avg      0.974     0.974     0.974     20601
weighted avg      0.975     0.974     0.974     20601


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.842     0.873     0.857      1466
         1.0      0.869     0.837     0.853      1477

    accuracy                          0.855      2943
   macro avg      0.855     0.855     0.855      2943
weighted avg      0.855     0.855     0.855      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.842     0.861     0.851      1303
         1.0      0.859     0.839     0.849      1314

    accuracy                          0.850      2617
   macro avg      0.850     0.850     0.850      2617
weighted avg      0.850     0.850     0.850      2617
```

SVM

[ ]:
```
from sklearn.svm import SVC

svm_cls = SVC(kernel = 'poly', degree = 3, gamma = 'auto', max_iter = 250,␣
 ↪verbose= True)
svm_cls.fit(x_train.reshape(n_train, -1), y_train)
preds_train = svm_cls.predict(x_train.reshape(n_train, -1))
preds_val   = svm_cls.predict(x_val.reshape(n_val, -1))
preds_test  = svm_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
[LibSVM]

c:\ProgramData\Anaconda3\envs\ml\lib\site-packages\sklearn\svm\_base.py:301:
ConvergenceWarning: Solver terminated early (max_iter=250).  Consider pre-
processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(

Training Classification Report:
               precision    recall  f1-score   support

         0.0      0.581     0.102     0.174     10260
         1.0      0.510     0.927     0.658     10341

    accuracy                          0.516     20601
   macro avg      0.545     0.515     0.416     20601
weighted avg      0.545     0.516     0.417     20601


Validation Classification Report:
               precision    recall  f1-score   support

         0.0      0.585     0.101     0.172      1466
         1.0      0.510     0.929     0.659      1477

    accuracy                          0.516      2943
   macro avg      0.548     0.515     0.415      2943
weighted avg      0.547     0.516     0.416      2943


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

```
         0.0       0.590     0.106     0.180      1303
         1.0       0.511     0.927     0.659      1314

    accuracy                           0.518      2617
   macro avg       0.550     0.516     0.419      2617
weighted avg       0.550     0.518     0.420      2617
```

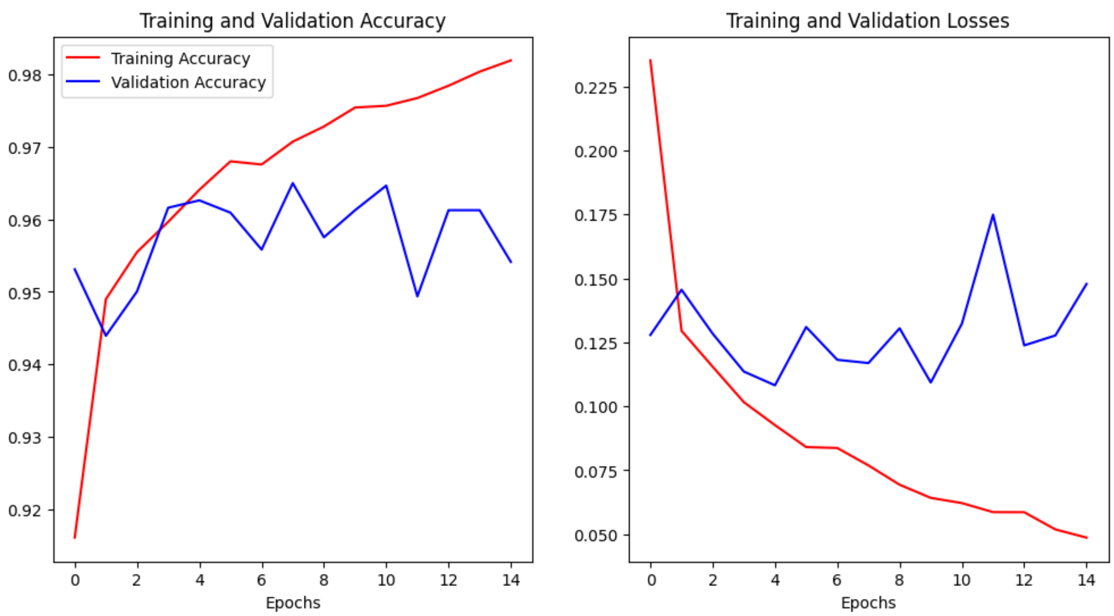
Transfer Learning

[ ]:
```
imagenet = tf.keras.applications.Xception(
    weights = 'imagenet',
    include_top = False,
    input_shape = (72, 72, 3)
)
imagenet.trainable = False

trans_learn = tf.keras.Sequential([
    tf.keras.layers.Resizing(72, 72),
    imagenet,

    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dense(2, activation = 'sigmoid')
])

trans_learn.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.003),
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)

history = trans_learn.fit(
            x_train, y_train, batch_size = 64,
            shuffle = True,
            epochs = 50,
            validation_data = [x_val, y_val],
            validation_batch_size = 32,
            callbacks = [tf.keras.callbacks.EarlyStopping(monitor = 'val_loss',␣
 ↪patience = 10),
                        tf.keras.callbacks.ModelCheckpoint(str = 'val_loss',␣
 ↪save_best_only = True, save_weights_only = True, filepath= "TL_No_Aug")]
        )
fig, ax = plot_history(history)

trans_learn.load_weights('TL_No_Aug')
```

```python
fig.savefig(PLOT_SAVE + "TL_unaug_losscurve.png", facecolor = 'white')
preds_train = np.argmax(trans_learn.predict(x_train), axis = 1)
preds_val   = np.argmax(trans_learn.predict(x_val), axis = 1)
preds_test  = np.argmax(trans_learn.predict(x_test), axis = 1)

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/50
322/322 [==============================] - 9s 19ms/step - loss: 0.2352 -
accuracy: 0.9161 - val_loss: 0.1279 - val_accuracy: 0.9531
Epoch 2/50
322/322 [==============================] - 5s 16ms/step - loss: 0.1295 -
accuracy: 0.9490 - val_loss: 0.1456 - val_accuracy: 0.9439
Epoch 3/50
322/322 [==============================] - 5s 16ms/step - loss: 0.1155 -
accuracy: 0.9555 - val_loss: 0.1284 - val_accuracy: 0.9501
Epoch 4/50
322/322 [==============================] - 5s 16ms/step - loss: 0.1016 -
accuracy: 0.9597 - val_loss: 0.1136 - val_accuracy: 0.9616
Epoch 5/50
322/322 [==============================] - 5s 16ms/step - loss: 0.0926 -
accuracy: 0.9641 - val_loss: 0.1082 - val_accuracy: 0.9626
Epoch 6/50
322/322 [==============================] - 5s 15ms/step - loss: 0.0841 -
accuracy: 0.9680 - val_loss: 0.1310 - val_accuracy: 0.9609
Epoch 7/50
322/322 [==============================] - 5s 15ms/step - loss: 0.0837 -
accuracy: 0.9676 - val_loss: 0.1182 - val_accuracy: 0.9558
Epoch 8/50
322/322 [==============================] - 5s 15ms/step - loss: 0.0769 -
accuracy: 0.9707 - val_loss: 0.1169 - val_accuracy: 0.9650
Epoch 9/50
322/322 [==============================] - 5s 15ms/step - loss: 0.0694 -
accuracy: 0.9728 - val_loss: 0.1305 - val_accuracy: 0.9575
Epoch 10/50
322/322 [==============================] - 5s 15ms/step - loss: 0.0642 -
accuracy: 0.9754 - val_loss: 0.1093 - val_accuracy: 0.9613
Epoch 11/50
322/322 [==============================] - 5s 15ms/step - loss: 0.0622 -
accuracy: 0.9757 - val_loss: 0.1323 - val_accuracy: 0.9647
Epoch 12/50
```

```
322/322 [==============================] - 5s 15ms/step - loss: 0.0587 -
accuracy: 0.9767 - val_loss: 0.1749 - val_accuracy: 0.9494
Epoch 13/50
322/322 [==============================] - 5s 15ms/step - loss: 0.0586 -
accuracy: 0.9784 - val_loss: 0.1238 - val_accuracy: 0.9613
Epoch 14/50
322/322 [==============================] - 5s 15ms/step - loss: 0.0519 -
accuracy: 0.9804 - val_loss: 0.1277 - val_accuracy: 0.9613
Epoch 15/50
322/322 [==============================] - 5s 15ms/step - loss: 0.0487 -
accuracy: 0.9819 - val_loss: 0.1478 - val_accuracy: 0.9541
```




```
644/644 [==============================] - 5s 7ms/step
92/92 [==============================] - 1s 7ms/step
82/82 [==============================] - 1s 7ms/step
Training Classification Report:
              precision    recall  f1-score   support

         0.0      0.959     0.985     0.972     10260
         1.0      0.985     0.958     0.971     10341

    accuracy                          0.972     20601
   macro avg      0.972     0.972     0.972     20601
weighted avg      0.972     0.972     0.972     20601


Validation Classification Report:
```

```
              precision    recall  f1-score   support

         0.0      0.952     0.974     0.963      1466
         1.0      0.974     0.951     0.962      1477

    accuracy                          0.963      2943
   macro avg      0.963     0.963     0.963      2943
weighted avg      0.963     0.963     0.963      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.944     0.968     0.956      1303
         1.0      0.967     0.943     0.955      1314

    accuracy                          0.955      2617
   macro avg      0.956     0.955     0.955      2617
weighted avg      0.956     0.955     0.955      2617
```

CNN Model

[ ]:
```
cnn = tf.keras.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), padding = 'same', activation = 'relu',␣
 ↪input_shape = x_train[0].shape),
    tf.keras.layers.MaxPool2D((3,3), padding = 'same'),

    tf.keras.layers.Conv2D(32, (2,2), padding = 'same', activation = 'relu'),
    tf.keras.layers.MaxPool2D((2,2), padding = 'same'),


    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(256, activation = 'relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(256, activation = 'relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(2, activation = 'sigmoid')
])

cnn.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.003),
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)

history = cnn.fit(
```

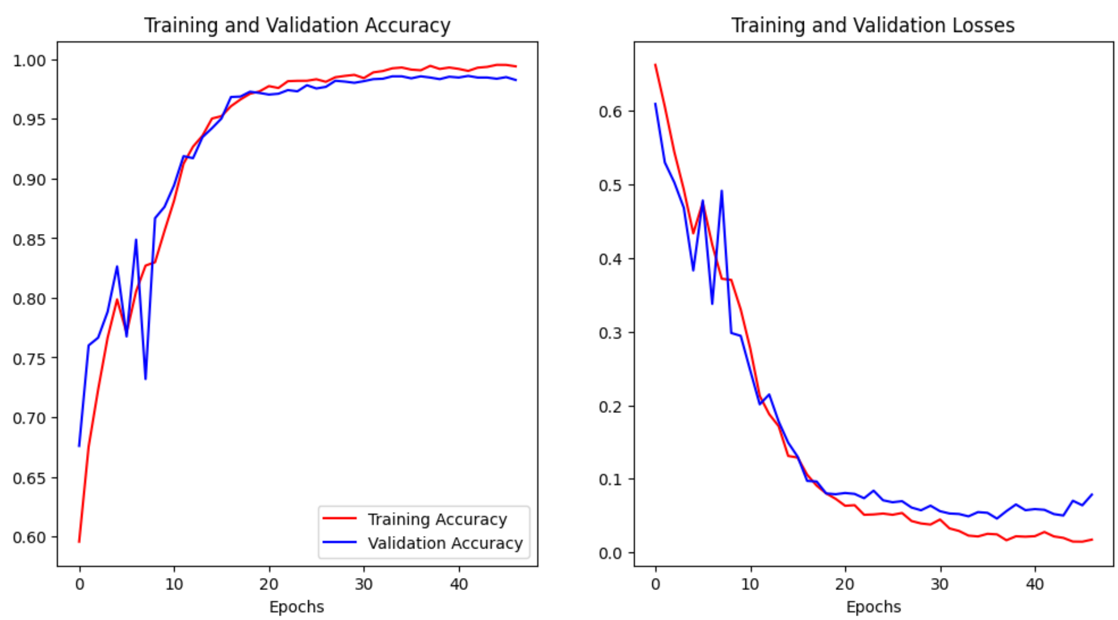
```
            x_train, y_train, batch_size = 1024,
            epochs = 50,
            validation_data = [x_val, y_val],
            validation_batch_size = 256,
            callbacks = [tf.keras.callbacks.EarlyStopping(monitor = 'val_loss',␣
↪patience = 10),
                         tf.keras.callbacks.ModelCheckpoint(str = 'val_loss',␣
↪save_best_only = True, save_weights_only = True, filepath= "CNN_No_Aug")]
        )


cnn.load_weights('CNN_No_Aug')
fig, ax = plot_history(history)

fig.savefig(PLOT_SAVE + "CNN_unaug_losscurve.png", facecolor = 'white')
preds_train = np.argmax(cnn.predict(x_train), axis = 1)
preds_val   = np.argmax(cnn.predict(x_val), axis = 1)
preds_test  = np.argmax(cnn.predict(x_test), axis = 1)

print("Training Classification Report: \n", classification_report(y_train,␣
↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/50
21/21 [==============================] - 1s 18ms/step - loss: 0.6625 - accuracy:
0.5956 - val_loss: 0.6094 - val_accuracy: 0.6758
Epoch 2/50
21/21 [==============================] - 0s 9ms/step - loss: 0.6057 - accuracy:
0.6752 - val_loss: 0.5299 - val_accuracy: 0.7601
Epoch 3/50
21/21 [==============================] - 0s 9ms/step - loss: 0.5441 - accuracy:
0.7230 - val_loss: 0.5027 - val_accuracy: 0.7666
Epoch 4/50
21/21 [==============================] - 0s 9ms/step - loss: 0.4931 - accuracy:
0.7666 - val_loss: 0.4683 - val_accuracy: 0.7883
Epoch 5/50
21/21 [==============================] - 0s 9ms/step - loss: 0.4335 - accuracy:
0.7987 - val_loss: 0.3831 - val_accuracy: 0.8264
Epoch 6/50
21/21 [==============================] - 0s 7ms/step - loss: 0.4756 - accuracy:
0.7706 - val_loss: 0.4782 - val_accuracy: 0.7676
Epoch 7/50
21/21 [==============================] - 0s 9ms/step - loss: 0.4180 - accuracy:
0.8054 - val_loss: 0.3376 - val_accuracy: 0.8488
```

```
Epoch 8/50
21/21 [==============================] - 0s 8ms/step - loss: 0.3718 - accuracy:
0.8270 - val_loss: 0.4914 - val_accuracy: 0.7319
Epoch 9/50
21/21 [==============================] - 0s 9ms/step - loss: 0.3703 - accuracy:
0.8299 - val_loss: 0.2982 - val_accuracy: 0.8668
Epoch 10/50
21/21 [==============================] - 0s 9ms/step - loss: 0.3300 - accuracy:
0.8563 - val_loss: 0.2942 - val_accuracy: 0.8763
Epoch 11/50
21/21 [==============================] - 0s 9ms/step - loss: 0.2771 - accuracy:
0.8814 - val_loss: 0.2473 - val_accuracy: 0.8943
Epoch 12/50
21/21 [==============================] - 0s 9ms/step - loss: 0.2127 - accuracy:
0.9124 - val_loss: 0.2012 - val_accuracy: 0.9188
Epoch 13/50
21/21 [==============================] - 0s 7ms/step - loss: 0.1878 - accuracy:
0.9268 - val_loss: 0.2146 - val_accuracy: 0.9171
Epoch 14/50
21/21 [==============================] - 0s 9ms/step - loss: 0.1714 - accuracy:
0.9360 - val_loss: 0.1777 - val_accuracy: 0.9348
Epoch 15/50
21/21 [==============================] - 0s 9ms/step - loss: 0.1311 - accuracy:
0.9503 - val_loss: 0.1491 - val_accuracy: 0.9422
Epoch 16/50
21/21 [==============================] - 0s 9ms/step - loss: 0.1288 - accuracy:
0.9523 - val_loss: 0.1300 - val_accuracy: 0.9504
Epoch 17/50
21/21 [==============================] - 0s 9ms/step - loss: 0.1055 - accuracy:
0.9607 - val_loss: 0.0972 - val_accuracy: 0.9684
Epoch 18/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0907 - accuracy:
0.9663 - val_loss: 0.0963 - val_accuracy: 0.9687
Epoch 19/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0802 - accuracy:
0.9711 - val_loss: 0.0801 - val_accuracy: 0.9728
Epoch 20/50
21/21 [==============================] - 0s 8ms/step - loss: 0.0728 - accuracy:
0.9731 - val_loss: 0.0790 - val_accuracy: 0.9718
Epoch 21/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0634 - accuracy:
0.9775 - val_loss: 0.0807 - val_accuracy: 0.9704
Epoch 22/50
21/21 [==============================] - 0s 8ms/step - loss: 0.0642 - accuracy:
0.9759 - val_loss: 0.0795 - val_accuracy: 0.9711
Epoch 23/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0511 - accuracy:
0.9817 - val_loss: 0.0736 - val_accuracy: 0.9742
```

```
Epoch 24/50
21/21 [==============================] - 0s 8ms/step - loss: 0.0516 - accuracy:
0.9819 - val_loss: 0.0839 - val_accuracy: 0.9732
Epoch 25/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0528 - accuracy:
0.9820 - val_loss: 0.0708 - val_accuracy: 0.9783
Epoch 26/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0511 - accuracy:
0.9833 - val_loss: 0.0681 - val_accuracy: 0.9755
Epoch 27/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0536 - accuracy:
0.9811 - val_loss: 0.0696 - val_accuracy: 0.9769
Epoch 28/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0426 - accuracy:
0.9850 - val_loss: 0.0609 - val_accuracy: 0.9820
Epoch 29/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0393 - accuracy:
0.9861 - val_loss: 0.0572 - val_accuracy: 0.9813
Epoch 30/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0378 - accuracy:
0.9869 - val_loss: 0.0635 - val_accuracy: 0.9803
Epoch 31/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0447 - accuracy:
0.9842 - val_loss: 0.0560 - val_accuracy: 0.9817
Epoch 32/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0326 - accuracy:
0.9890 - val_loss: 0.0529 - val_accuracy: 0.9834
Epoch 33/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0291 - accuracy:
0.9901 - val_loss: 0.0522 - val_accuracy: 0.9837
Epoch 34/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0228 - accuracy:
0.9924 - val_loss: 0.0490 - val_accuracy: 0.9857
Epoch 35/50
21/21 [==============================] - 0s 8ms/step - loss: 0.0216 - accuracy:
0.9931 - val_loss: 0.0548 - val_accuracy: 0.9857
Epoch 36/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0252 - accuracy:
0.9914 - val_loss: 0.0538 - val_accuracy: 0.9840
Epoch 37/50
21/21 [==============================] - 0s 9ms/step - loss: 0.0245 - accuracy:
0.9908 - val_loss: 0.0459 - val_accuracy: 0.9857
Epoch 38/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0165 - accuracy:
0.9945 - val_loss: 0.0561 - val_accuracy: 0.9847
Epoch 39/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0219 - accuracy:
0.9919 - val_loss: 0.0651 - val_accuracy: 0.9834
```

```
Epoch 40/50
21/21 [==============================] - 0s 8ms/step - loss: 0.0213 - accuracy:
0.9931 - val_loss: 0.0574 - val_accuracy: 0.9854
Epoch 41/50
21/21 [==============================] - 0s 8ms/step - loss: 0.0220 - accuracy:
0.9919 - val_loss: 0.0588 - val_accuracy: 0.9847
Epoch 42/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0277 - accuracy:
0.9902 - val_loss: 0.0579 - val_accuracy: 0.9861
Epoch 43/50
21/21 [==============================] - 0s 8ms/step - loss: 0.0217 - accuracy:
0.9930 - val_loss: 0.0520 - val_accuracy: 0.9847
Epoch 44/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0197 - accuracy:
0.9936 - val_loss: 0.0501 - val_accuracy: 0.9847
Epoch 45/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0147 - accuracy:
0.9953 - val_loss: 0.0702 - val_accuracy: 0.9837
Epoch 46/50
21/21 [==============================] - 0s 7ms/step - loss: 0.0146 - accuracy:
0.9953 - val_loss: 0.0640 - val_accuracy: 0.9850
Epoch 47/50
21/21 [==============================] - 0s 8ms/step - loss: 0.0173 - accuracy:
0.9941 - val_loss: 0.0785 - val_accuracy: 0.9827
```




```
644/644 [==============================] - 1s 1ms/step
92/92 [==============================] - 0s 1ms/step
```

```
82/82 [==============================] - 0s 1ms/step
Training Classification Report:
               precision    recall  f1-score   support

         0.0      0.994     0.997     0.996     10260
         1.0      0.997     0.994     0.996     10341

    accuracy                          0.996     20601
   macro avg      0.996     0.996     0.996     20601
weighted avg      0.996     0.996     0.996     20601


Validation Classification Report:
               precision    recall  f1-score   support

         0.0      0.983     0.988     0.986      1466
         1.0      0.988     0.983     0.986      1477

    accuracy                          0.986      2943
   macro avg      0.986     0.986     0.986      2943
weighted avg      0.986     0.986     0.986      2943


Testing Classification Report:
               precision    recall  f1-score   support

         0.0      0.976     0.988     0.982      1303
         1.0      0.988     0.976     0.982      1314

    accuracy                          0.982      2617
   macro avg      0.982     0.982     0.982      2617
weighted avg      0.982     0.982     0.982      2617
```

0.6.2 Augmented Dataset

Naive Bayes

[ ]:
```
from sklearn import naive_bayes

nb_cls = naive_bayes.GaussianNB()
nb_cls.fit(x_train_aug.reshape(n_aug_train, -1), y_train_aug)
preds_train = nb_cls.predict(x_train_aug.reshape(n_aug_train, -1))
preds_val   = nb_cls.predict(x_val.reshape(n_val, -1))
preds_test  = nb_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train_aug,␣
↪preds_train, digits = N_DIGITS))
```

```python
print("\nValidation Classification Report: \n", classification_report(y_val,␣
↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
↪preds_test, digits = N_DIGITS))
```

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

         0.0      0.607     0.760     0.675     20520
         1.0      0.683     0.512     0.586     20682

    accuracy                          0.636     41202
   macro avg      0.645     0.636     0.631     41202
weighted avg      0.645     0.636     0.630     41202


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1466
         1.0      0.502     1.000     0.668      1477

    accuracy                          0.502      2943
   macro avg      0.251     0.500     0.334      2943
weighted avg      0.252     0.502     0.335      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1303
         1.0      0.502     1.000     0.669      1314

    accuracy                          0.502      2617
   macro avg      0.251     0.500     0.334      2617
weighted avg      0.252     0.502     0.336      2617

c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
```

```
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
```

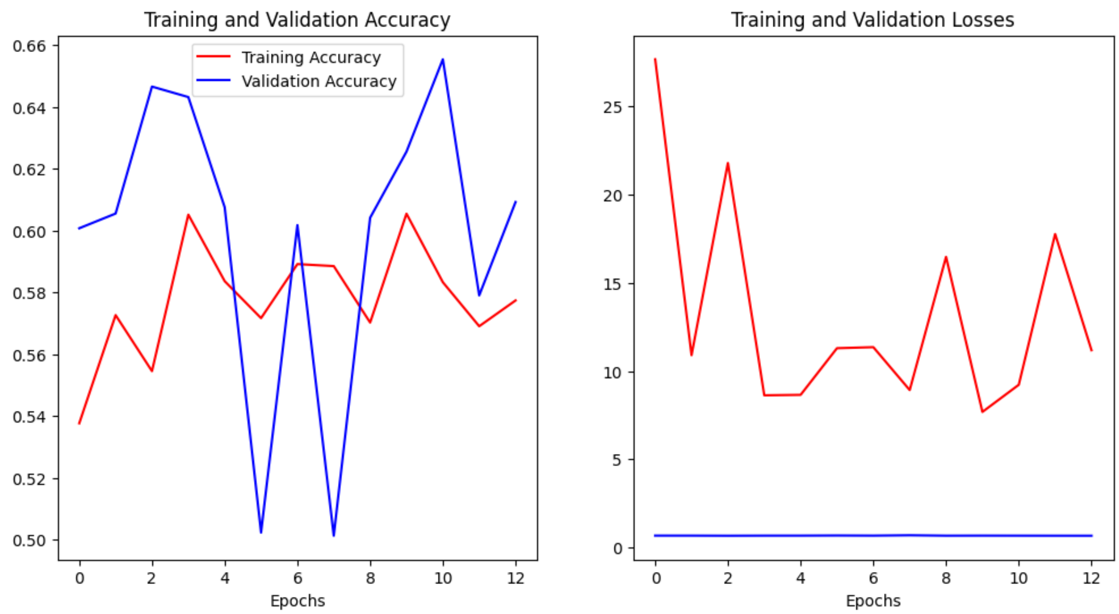
Logistic Regression

[ ]:
```
from sklearn.linear_model import LogisticRegression

logreg_cls =  tf.keras.Sequential([tf.keras.layers.Dense(2, activation =
 ↪'sigmoid', input_dim = HEIGHT * WIDTH * 3)])
logreg_cls.compile(optimizer = tf.optimizers.Adam(0.001), loss =
 ↪'sparse_categorical_crossentropy', metrics = ['accuracy'])
history = logreg_cls.fit(x_train_aug.reshape(n_aug_train, -1),
                         y_train_aug,
                         batch_size = 512,
                         validation_data=[x_val.reshape(n_val, -1), y_val],
                         validation_batch_size=128,
                         epochs = 100,
                         callbacks =  [tf.keras.callbacks.EarlyStopping(monitor
 ↪= 'val_loss', patience = 10),
                                      tf.keras.callbacks.ModelCheckpoint(str =
 ↪'val_loss', save_best_only = True, save_weights_only = True, filepath=
 ↪"LR_Aug")])
preds_train = np.argmax(logreg_cls.predict(x_train_aug.reshape(n_aug_train,
 ↪-1)), axis = 1)
preds_val   = np.argmax(logreg_cls.predict(x_val.reshape(n_val, -1)), axis = 1)
preds_test  = np.argmax(logreg_cls.predict(x_test.reshape(n_test, -1)), axis =
 ↪1)

logreg_cls.load_weights('LR_Aug')
fig, ax = plot_history(history)
```

```
fig.savefig(PLOT_SAVE + "logreg_aug_losscurve.png", facecolor = 'white')

print("Training Classification Report: \n", classification_report(y_train_aug,␣
↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/100
81/81 [==============================] - 0s 3ms/step - loss: 27.6512 - accuracy:
0.5376 - val_loss: 0.6819 - val_accuracy: 0.6007
Epoch 2/100
81/81 [==============================] - 0s 2ms/step - loss: 10.8976 - accuracy:
0.5726 - val_loss: 0.6814 - val_accuracy: 0.6055
Epoch 3/100
81/81 [==============================] - 0s 2ms/step - loss: 21.7795 - accuracy:
0.5545 - val_loss: 0.6752 - val_accuracy: 0.6466
Epoch 4/100
81/81 [==============================] - 0s 2ms/step - loss: 8.6269 - accuracy:
0.6051 - val_loss: 0.6807 - val_accuracy: 0.6432
Epoch 5/100
81/81 [==============================] - 0s 2ms/step - loss: 8.6525 - accuracy:
0.5836 - val_loss: 0.6813 - val_accuracy: 0.6075
Epoch 6/100
81/81 [==============================] - 0s 3ms/step - loss: 11.2940 - accuracy:
0.5716 - val_loss: 0.6885 - val_accuracy: 0.5022
Epoch 7/100
81/81 [==============================] - 0s 2ms/step - loss: 11.3517 - accuracy:
0.5891 - val_loss: 0.6810 - val_accuracy: 0.6018
Epoch 8/100
81/81 [==============================] - 0s 2ms/step - loss: 8.9236 - accuracy:
0.5885 - val_loss: 0.7014 - val_accuracy: 0.5012
Epoch 9/100
81/81 [==============================] - 0s 2ms/step - loss: 16.4641 - accuracy:
0.5702 - val_loss: 0.6786 - val_accuracy: 0.6041
Epoch 10/100
81/81 [==============================] - 0s 2ms/step - loss: 7.6863 - accuracy:
0.6055 - val_loss: 0.6820 - val_accuracy: 0.6256
Epoch 11/100
81/81 [==============================] - 0s 2ms/step - loss: 9.2223 - accuracy:
0.5833 - val_loss: 0.6793 - val_accuracy: 0.6555
Epoch 12/100
81/81 [==============================] - 0s 2ms/step - loss: 17.7575 - accuracy:
0.5690 - val_loss: 0.6769 - val_accuracy: 0.5790
Epoch 13/100
```

```
81/81 [==============================] - 0s 3ms/step - loss: 11.1797 - accuracy:
0.5774 - val_loss: 0.6755 - val_accuracy: 0.6092
1288/1288 [==============================] - 1s 712us/step
92/92 [==============================] - 0s 769us/step
82/82 [==============================] - 0s 784us/step
```




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

         0.0      0.638     0.430     0.514     20520
         1.0      0.573     0.759     0.653     20682

    accuracy                          0.595     41202
   macro avg      0.606     0.594     0.583     41202
weighted avg      0.605     0.595     0.583     41202


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.813     0.280     0.416      1466
         1.0      0.567     0.936     0.706      1477

    accuracy                          0.609      2943
   macro avg      0.690     0.608     0.561      2943
weighted avg      0.690     0.609     0.562      2943
```

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

         0.0      0.776     0.243     0.370      1303
         1.0      0.553     0.931     0.694      1314

    accuracy                          0.588      2617
   macro avg      0.665     0.587     0.532      2617
weighted avg      0.664     0.588     0.533      2617
```

Decision Trees

```
[ ]: from sklearn.tree import DecisionTreeClassifier

dt_cls =  DecisionTreeClassifier(criterion = 'gini', splitter = 'best',␣
 ↪max_depth = 5, min_samples_split = 2, )
dt_cls.fit(x_train_aug.reshape(n_aug_train, -1), y_train_aug)
preds_train = dt_cls.predict(x_train_aug.reshape(n_aug_train, -1))
preds_val   = dt_cls.predict(x_val.reshape(n_val, -1))
preds_test  = dt_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train_aug,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))

# eval = testing_module.ModelEvaluation(y_val,
#     nb_cls.predict(x_val.reshape(n_val, -1)),
#     model_reference_name = 'Naive Bayes',
#     model_type = 'classification',
#     plot_classification_metric = ['roc_auc']) # if classification
# eval.evaluate(evaluate_save= True, plots_show = True)
```

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

         0.0      0.687     0.768     0.725     20520
         1.0      0.739     0.653     0.693     20682

    accuracy                          0.710     41202
   macro avg      0.713     0.710     0.709     41202
weighted avg      0.713     0.710     0.709     41202


Validation Classification Report:
```

```
              precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1466
         1.0      0.502     1.000     0.668      1477

    accuracy                          0.502      2943
   macro avg      0.251     0.500     0.334      2943
weighted avg      0.252     0.502     0.335      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1303
         1.0      0.502     1.000     0.669      1314

    accuracy                          0.502      2617
   macro avg      0.251     0.500     0.334      2617
weighted avg      0.252     0.502     0.336      2617

c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
```

```
Precision and F-score are 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, msg_start, len(result))
```

XGBoost

```
[ ]: from xgboost import XGBClassifier

xgb_cls = XGBClassifier(max_depth = 5, objective = 'reg:logistic',
                            num_parallel_tree = 20, booster = 'gbtree',
                            gamma = 0.5, tree_method = 'gpu_hist', subsample =␣
 ↪0.4, reg_lambda = 1)
xgb_cls.fit(x_train_aug.reshape(n_aug_train, -1), y_train_aug)
preds_train = xgb_cls.predict(x_train_aug.reshape(n_aug_train, -1))
preds_val   = xgb_cls.predict(x_val.reshape(n_val, -1))
preds_test  = xgb_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train_aug,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

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

         0.0      0.915     0.954     0.934     20520
         1.0      0.953     0.912     0.932     20682

    accuracy                          0.933     41202
   macro avg      0.934     0.933     0.933     41202
weighted avg      0.934     0.933     0.933     41202


Validation Classification Report:
               precision    recall  f1-score   support

         0.0      0.498     1.000     0.665      1466
         1.0      0.000     0.000     0.000      1477

    accuracy                          0.498      2943
   macro avg      0.249     0.500     0.333      2943
weighted avg      0.248     0.498     0.331      2943


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

```
         0.0      0.498     1.000     0.665      1303
         1.0      0.000     0.000     0.000      1314

    accuracy                          0.498      2617
   macro avg      0.249     0.500     0.332      2617
weighted avg      0.248     0.498     0.331      2617
```

```
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
```

SVM

[ ]:
```
from sklearn.svm import SVC

svm_cls = SVC(kernel = 'rbf', max_iter = 250, verbose= True)
svm_cls.fit(x_train_aug.reshape(n_aug_train, -1), y_train_aug)
preds_train = svm_cls.predict(x_train_aug.reshape(n_aug_train, -1))
preds_val   = svm_cls.predict(x_val.reshape(n_val, -1))
preds_test  = svm_cls.predict(x_test.reshape(n_test, -1))
```

```
print("Training Classification Report: \n", classification_report(y_train_aug,␣
↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
↪preds_test, digits = N_DIGITS))
```

```
[LibSVM]

c:\ProgramData\Anaconda3\envs\ml\lib\site-packages\sklearn\svm\_base.py:301:
ConvergenceWarning: Solver terminated early (max_iter=250).  Consider pre-
processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(

Training Classification Report:
               precision    recall  f1-score   support

         0.0      0.449     0.132     0.204     20520
         1.0      0.494     0.840     0.622     20682

    accuracy                          0.487     41202
   macro avg      0.472     0.486     0.413     41202
weighted avg      0.472     0.487     0.414     41202


Validation Classification Report:
               precision    recall  f1-score   support

         0.0      0.498     1.000     0.665      1466
         1.0      0.000     0.000     0.000      1477

    accuracy                          0.498      2943
   macro avg      0.249     0.500     0.333      2943
weighted avg      0.248     0.498     0.331      2943


Testing Classification Report:
               precision    recall  f1-score   support

         0.0      0.498     1.000     0.665      1303
         1.0      0.000     0.000     0.000      1314

    accuracy                          0.498      2617
   macro avg      0.249     0.500     0.332      2617
weighted avg      0.248     0.498     0.331      2617
```

```
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
```

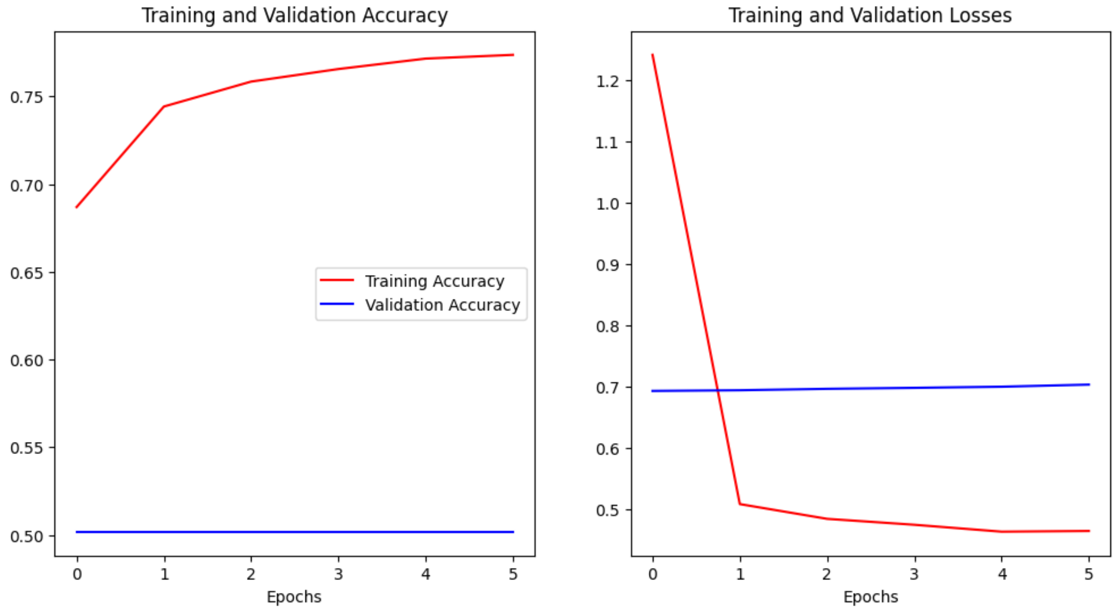
Transfer Learning

[ ]:
```python
imagenet = tf.keras.applications.Xception(
    weights = 'imagenet',
    include_top = False,
    input_shape = (72, 72, 3)
)
imagenet.trainable = False

trans_learn = tf.keras.Sequential([
    tf.keras.layers.Resizing(72, 72),
    imagenet,

    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dense(2, activation = 'sigmoid')
```

```python
])

trans_learn.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.001),
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)

history = trans_learn.fit(
            x_train_aug, y_train_aug, batch_size = 128,
            shuffle = True,
            epochs = 50,
            validation_data = [x_val, y_val],
            validation_batch_size = 64,
            callbacks = [tf.keras.callbacks.EarlyStopping(monitor = 'val_loss',␣
 ↪patience = 5),
                        tf.keras.callbacks.ModelCheckpoint(str = 'val_loss',␣
 ↪save_best_only = True, save_weights_only = True, filepath= "TL_Aug")]
        )

trans_learn.load_weights('TL_Aug')

fig, ax = plot_history(history)

fig.savefig(PLOT_SAVE + "TL_aug_losscurve.png", facecolor = 'white')
preds_train = np.argmax(trans_learn.predict(x_train_aug), axis = 1)
preds_val   = np.argmax(trans_learn.predict(x_val), axis = 1)
preds_test  = np.argmax(trans_learn.predict(x_test), axis = 1)

print("Training Classification Report: \n", classification_report(y_train_aug,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/50
322/322 [==============================] - 11s 27ms/step - loss: 1.2411 -
accuracy: 0.6869 - val_loss: 0.6934 - val_accuracy: 0.5019
Epoch 2/50
322/322 [==============================] - 8s 24ms/step - loss: 0.5090 -
accuracy: 0.7442 - val_loss: 0.6945 - val_accuracy: 0.5019
Epoch 3/50
322/322 [==============================] - 8s 24ms/step - loss: 0.4849 -
accuracy: 0.7584 - val_loss: 0.6968 - val_accuracy: 0.5019
Epoch 4/50
322/322 [==============================] - 8s 24ms/step - loss: 0.4753 -
```

```
accuracy: 0.7656 - val_loss: 0.6984 - val_accuracy: 0.5019
Epoch 5/50
322/322 [==============================] - 8s 24ms/step - loss: 0.4640 -
accuracy: 0.7715 - val_loss: 0.7003 - val_accuracy: 0.5019
Epoch 6/50
322/322 [==============================] - 8s 24ms/step - loss: 0.4652 -
accuracy: 0.7737 - val_loss: 0.7037 - val_accuracy: 0.5019
```




```
1288/1288 [==============================] - 9s 7ms/step
92/92 [==============================] - 1s 7ms/step
82/82 [==============================] - 1s 7ms/step
Training Classification Report:
              precision    recall  f1-score   support

         0.0      0.765     0.754     0.760     20520
         1.0      0.760     0.770     0.765     20682

    accuracy                          0.762     41202
   macro avg      0.762     0.762     0.762     41202
weighted avg      0.762     0.762     0.762     41202


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1466
         1.0      0.502     1.000     0.668      1477
```

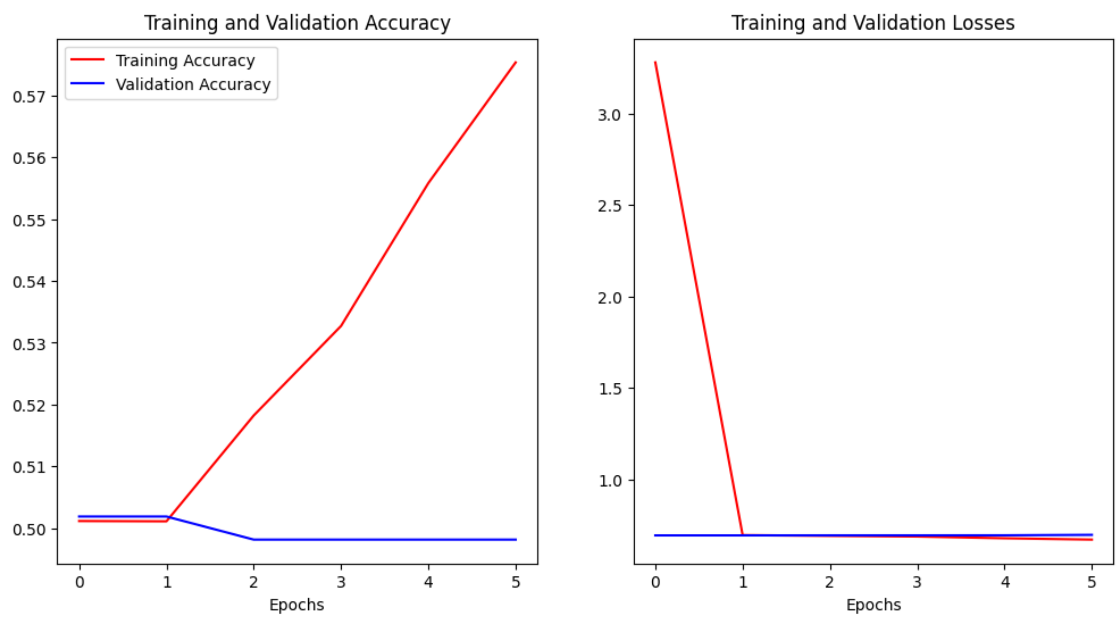
```
    accuracy                         0.502      2943
   macro avg      0.251     0.500     0.334      2943
weighted avg      0.252     0.502     0.335      2943


Testing Classification Report:
               precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1303
         1.0      0.502     1.000     0.669      1314

    accuracy                          0.502      2617
   macro avg      0.251     0.500     0.334      2617
weighted avg      0.252     0.502     0.336      2617

c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
```

CNN Model

[ ]:
```
cnn = tf.keras.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), padding = 'same', activation = 'relu',␣
 ↪input_shape = x_train[0].shape),
    tf.keras.layers.MaxPool2D((3,3), padding = 'same'),

    tf.keras.layers.Conv2D(32, (2,2), padding = 'same', activation = 'relu'),
    tf.keras.layers.MaxPool2D((2,2), padding = 'same'),


    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(256, activation = 'relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(256, activation = 'relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(2, activation = 'sigmoid')
])

cnn.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.001),
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)

history = cnn.fit(
            x_train_aug, y_train_aug, batch_size = 256,
            epochs = 50,
            shuffle = True,
            validation_data = [x_val, y_val],
            validation_batch_size = 128,
            callbacks = [
                tf.keras.callbacks.EarlyStopping(monitor = 'val_loss', patience␣
 ↪= 5),
                tf.keras.callbacks.ModelCheckpoint(str = 'val_loss',␣
 ↪save_best_only = True, save_weights_only = True, filepath= "CNN_Aug")
            ]
        )

fig, ax = plot_history(history)

cnn.load_weights('CNN_Aug')

fig.savefig(PLOT_SAVE + "CNN_aug_losscurve.png", facecolor = 'white')
preds_train = np.argmax(cnn.predict(x_train_aug), axis = 1)
preds_val   = np.argmax(cnn.predict(x_val), axis = 1)
preds_test  = np.argmax(cnn.predict(x_test), axis = 1)
```

```
print("Training Classification Report: \n", classification_report(y_train_aug,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/50
161/161 [==============================] - 1s 4ms/step - loss: 3.2800 -
accuracy: 0.5011 - val_loss: 0.6931 - val_accuracy: 0.5019
Epoch 2/50
161/161 [==============================] - 1s 3ms/step - loss: 0.6945 -
accuracy: 0.5011 - val_loss: 0.6932 - val_accuracy: 0.5019
Epoch 3/50
161/161 [==============================] - 1s 3ms/step - loss: 0.6905 -
accuracy: 0.5182 - val_loss: 0.6933 - val_accuracy: 0.4981
Epoch 4/50
161/161 [==============================] - 1s 3ms/step - loss: 0.6869 -
accuracy: 0.5327 - val_loss: 0.6934 - val_accuracy: 0.4981
Epoch 5/50
161/161 [==============================] - 1s 3ms/step - loss: 0.6775 -
accuracy: 0.5558 - val_loss: 0.6937 - val_accuracy: 0.4981
Epoch 6/50
161/161 [==============================] - 1s 3ms/step - loss: 0.6701 -
accuracy: 0.5754 - val_loss: 0.6960 - val_accuracy: 0.4981
```



```
1288/1288 [==============================] - 1s 1ms/step
92/92 [==============================] - 0s 1ms/step
82/82 [==============================] - 0s 1ms/step
Training Classification Report:
              precision    recall  f1-score   support

         0.0      0.487     0.011     0.021     20520
         1.0      0.502     0.989     0.666     20682

    accuracy                          0.502     41202
   macro avg      0.494     0.500     0.344     41202
weighted avg      0.494     0.502     0.345     41202


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1466
         1.0      0.502     1.000     0.668      1477

    accuracy                          0.502      2943
   macro avg      0.251     0.500     0.334      2943
weighted avg      0.252     0.502     0.335      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1303
         1.0      0.502     1.000     0.669      1314

    accuracy                          0.502      2617
   macro avg      0.251     0.500     0.334      2617
weighted avg      0.252     0.502     0.336      2617

c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
```

```
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
```

0.7 HSV Modelling with only the saturation dimension

Transfer learning does not work since it requires exactly 3 input channels (Xception, VGG19, etc)

0.7.1 Preparing Datasets

[ ]:
```
temp = []
for img in tqdm(x_train):
    temp.append(rgb2hsv(img))
temp = np.array(temp)[..., 1]
x_train = temp
x_train = x_train[..., np.newaxis]

temp = []
for img in tqdm(x_val):
    temp.append(rgb2hsv(img))
temp = np.array(temp)[..., 1]
x_val = temp
x_val = x_val[..., np.newaxis]

temp = []
for img in tqdm(x_test):
    temp.append(rgb2hsv(img))
temp = np.array(temp)[..., 1]
x_test = temp
x_test = x_test[..., np.newaxis]



temp = []
```

```
for img in tqdm(x_train_aug):
    temp.append(rgb2hsv(img))
temp = np.array(temp)[..., 1]
x_train_aug = temp
x_train_aug = x_train_aug[..., np.newaxis]



x_train.shape, x_val.shape, x_test.shape
```

```
100%|      | 20601/20601 [00:03<00:00, 6839.57it/s]
100%|      | 2943/2943 [00:00<00:00, 6781.57it/s]
100%|      | 2617/2617 [00:00<00:00, 6946.58it/s]
100%|      | 41202/41202 [00:06<00:00, 6203.58it/s]
```

[ ]: ((20601, 25, 25, 1), (2943, 25, 25, 1), (2617, 25, 25, 1))

0.7.2 Unaugmented

Naive Bayes

[ ]:
```
from sklearn import naive_bayes

nb_cls = naive_bayes.GaussianNB()
nb_cls.fit(x_train.reshape(n_train, -1), y_train)
preds_train = nb_cls.predict(x_train.reshape(n_train, -1))
preds_val   = nb_cls.predict(x_val.reshape(n_val, -1))
preds_test  = nb_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))

# eval = testing_module.ModelEvaluation(y_val,
#     nb_cls.predict(x_val.reshape(n_val, -1)),
#     model_reference_name = 'Naive Bayes',
#     model_type = 'classification',
#     plot_classification_metric = ['roc_auc']) # if classification
# eval.evaluate(evaluate_save= True, plots_show = True)
```

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

         0.0      0.607     0.565     0.585     10260
         1.0      0.596     0.636     0.615     10341
```

```
    accuracy                         0.601     20601
   macro avg      0.601     0.601     0.600     20601
weighted avg      0.601     0.601     0.600     20601


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.604     0.586     0.595      1466
         1.0      0.601     0.618     0.609      1477

    accuracy                          0.602      2943
   macro avg      0.602     0.602     0.602      2943
weighted avg      0.602     0.602     0.602      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.632     0.585     0.607      1303
         1.0      0.617     0.662     0.639      1314

    accuracy                          0.624      2617
   macro avg      0.624     0.623     0.623      2617
weighted avg      0.624     0.624     0.623      2617
```

Logistic Regression

[ ]:
```
from sklearn.linear_model import LogisticRegression

logreg_cls =  tf.keras.Sequential([tf.keras.layers.Dense(2, activation =
 ↪'sigmoid', input_dim = HEIGHT * WIDTH)])
logreg_cls.compile(optimizer = tf.optimizers.Adam(0.002), loss =
 ↪'sparse_categorical_crossentropy', metrics = ['accuracy'])
history = logreg_cls.fit(x_train.reshape(n_train, -1),
                        y_train,
                        batch_size = 512,
                        validation_data=[x_val.reshape(n_val, -1), y_val],
                        validation_batch_size=128,
                        epochs = 100,
                        callbacks =  [tf.keras.callbacks.EarlyStopping(monitor
 ↪= 'val_loss', patience = 10),
                                      tf.keras.callbacks.ModelCheckpoint(str =
 ↪'val_loss', save_best_only = True, save_weights_only = True, filepath=
 ↪"LR_No_Aug_HSV")],
                        verbose = 1)
```

```
logreg_cls.load_weights('LR_No_Aug_HSV')
preds_train = np.argmax(logreg_cls.predict(x_train.reshape(n_train, -1)), axis␣
 ↪= 1)
preds_val   = np.argmax(logreg_cls.predict(x_val.reshape(n_val, -1)), axis = 1)
preds_test  = np.argmax(logreg_cls.predict(x_test.reshape(n_test, -1)), axis =␣
 ↪1)

fig, ax = plot_history(history)

fig.savefig(PLOT_SAVE + "logreg_unaug_losscurve_HSV.png", facecolor = 'white')

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/100
41/41 [==============================] - 1s 6ms/step - loss: 0.6936 - accuracy:
0.5144 - val_loss: 0.6743 - val_accuracy: 0.5260
Epoch 2/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6704 - accuracy:
0.5738 - val_loss: 0.6610 - val_accuracy: 0.5929
Epoch 3/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6579 - accuracy:
0.6304 - val_loss: 0.6498 - val_accuracy: 0.6521
Epoch 4/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6484 - accuracy:
0.6475 - val_loss: 0.6431 - val_accuracy: 0.6677
Epoch 5/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6426 - accuracy:
0.6536 - val_loss: 0.6367 - val_accuracy: 0.6558
Epoch 6/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6360 - accuracy:
0.6585 - val_loss: 0.6295 - val_accuracy: 0.6701
Epoch 7/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6309 - accuracy:
0.6639 - val_loss: 0.6249 - val_accuracy: 0.6765
Epoch 8/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6281 - accuracy:
0.6604 - val_loss: 0.6234 - val_accuracy: 0.6660
Epoch 9/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6248 - accuracy:
0.6643 - val_loss: 0.6195 - val_accuracy: 0.6762
Epoch 10/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6213 - accuracy:
```

```
0.6685 - val_loss: 0.6161 - val_accuracy: 0.6704
Epoch 11/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6190 - accuracy:
0.6676 - val_loss: 0.6161 - val_accuracy: 0.6724
Epoch 12/100
41/41 [==============================] - 0s 4ms/step - loss: 0.6192 - accuracy:
0.6609 - val_loss: 0.6135 - val_accuracy: 0.6721
Epoch 13/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6175 - accuracy:
0.6661 - val_loss: 0.6260 - val_accuracy: 0.6463
Epoch 14/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6166 - accuracy:
0.6676 - val_loss: 0.6092 - val_accuracy: 0.6762
Epoch 15/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6146 - accuracy:
0.6663 - val_loss: 0.6089 - val_accuracy: 0.6680
Epoch 16/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6123 - accuracy:
0.6716 - val_loss: 0.6089 - val_accuracy: 0.6639
Epoch 17/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6114 - accuracy:
0.6711 - val_loss: 0.6149 - val_accuracy: 0.6619
Epoch 18/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6115 - accuracy:
0.6698 - val_loss: 0.6057 - val_accuracy: 0.6762
Epoch 19/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6101 - accuracy:
0.6714 - val_loss: 0.6049 - val_accuracy: 0.6741
Epoch 20/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6075 - accuracy:
0.6748 - val_loss: 0.6042 - val_accuracy: 0.6724
Epoch 21/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6077 - accuracy:
0.6744 - val_loss: 0.6047 - val_accuracy: 0.6789
Epoch 22/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6070 - accuracy:
0.6755 - val_loss: 0.6041 - val_accuracy: 0.6687
Epoch 23/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6056 - accuracy:
0.6775 - val_loss: 0.6025 - val_accuracy: 0.6735
Epoch 24/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6058 - accuracy:
0.6781 - val_loss: 0.6038 - val_accuracy: 0.6769
Epoch 25/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6059 - accuracy:
0.6766 - val_loss: 0.6027 - val_accuracy: 0.6772
Epoch 26/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6059 - accuracy:
```

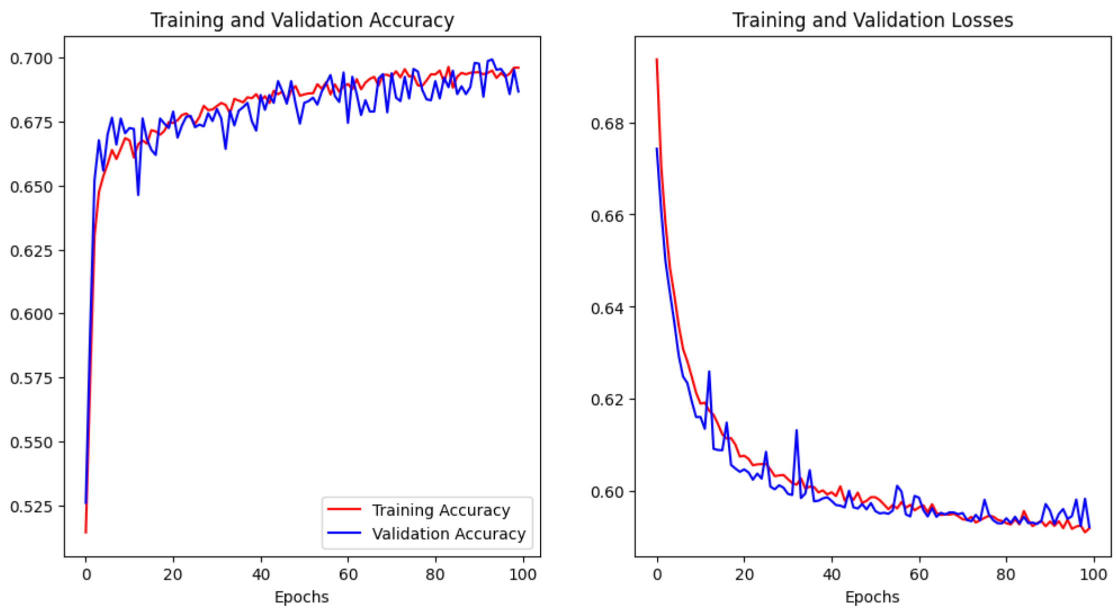
```
0.6740 - val_loss: 0.6086 - val_accuracy: 0.6728
Epoch 27/100
41/41 [==============================] - 0s 4ms/step - loss: 0.6047 - accuracy:
0.6768 - val_loss: 0.6010 - val_accuracy: 0.6738
Epoch 28/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6033 - accuracy:
0.6811 - val_loss: 0.6004 - val_accuracy: 0.6731
Epoch 29/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6034 - accuracy:
0.6794 - val_loss: 0.6013 - val_accuracy: 0.6782
Epoch 30/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6035 - accuracy:
0.6797 - val_loss: 0.6008 - val_accuracy: 0.6752
Epoch 31/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6026 - accuracy:
0.6810 - val_loss: 0.5995 - val_accuracy: 0.6799
Epoch 32/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6018 - accuracy:
0.6822 - val_loss: 0.5992 - val_accuracy: 0.6762
Epoch 33/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6014 - accuracy:
0.6815 - val_loss: 0.6132 - val_accuracy: 0.6643
Epoch 34/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6028 - accuracy:
0.6782 - val_loss: 0.5985 - val_accuracy: 0.6792
Epoch 35/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6006 - accuracy:
0.6839 - val_loss: 0.5995 - val_accuracy: 0.6735
Epoch 36/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6010 - accuracy:
0.6832 - val_loss: 0.6046 - val_accuracy: 0.6792
Epoch 37/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6009 - accuracy:
0.6825 - val_loss: 0.5978 - val_accuracy: 0.6806
Epoch 38/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5998 - accuracy:
0.6845 - val_loss: 0.5980 - val_accuracy: 0.6823
Epoch 39/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6002 - accuracy:
0.6841 - val_loss: 0.5985 - val_accuracy: 0.6752
Epoch 40/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5993 - accuracy:
0.6857 - val_loss: 0.5987 - val_accuracy: 0.6714
Epoch 41/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5998 - accuracy:
0.6830 - val_loss: 0.5979 - val_accuracy: 0.6854
Epoch 42/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5990 - accuracy:
```

```
0.6849 - val_loss: 0.5970 - val_accuracy: 0.6796
Epoch 43/100
41/41 [==============================] - 0s 3ms/step - loss: 0.6011 - accuracy:
0.6822 - val_loss: 0.5969 - val_accuracy: 0.6854
Epoch 44/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5980 - accuracy:
0.6870 - val_loss: 0.5965 - val_accuracy: 0.6823
Epoch 45/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5992 - accuracy:
0.6856 - val_loss: 0.6001 - val_accuracy: 0.6908
Epoch 46/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5981 - accuracy:
0.6867 - val_loss: 0.5965 - val_accuracy: 0.6867
Epoch 47/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5997 - accuracy:
0.6827 - val_loss: 0.5963 - val_accuracy: 0.6820
Epoch 48/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5975 - accuracy:
0.6870 - val_loss: 0.5972 - val_accuracy: 0.6908
Epoch 49/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5979 - accuracy:
0.6889 - val_loss: 0.5961 - val_accuracy: 0.6820
Epoch 50/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5987 - accuracy:
0.6851 - val_loss: 0.5975 - val_accuracy: 0.6741
Epoch 51/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5987 - accuracy:
0.6856 - val_loss: 0.5957 - val_accuracy: 0.6823
Epoch 52/100
41/41 [==============================] - 0s 4ms/step - loss: 0.5981 - accuracy:
0.6860 - val_loss: 0.5952 - val_accuracy: 0.6830
Epoch 53/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5971 - accuracy:
0.6860 - val_loss: 0.5953 - val_accuracy: 0.6843
Epoch 54/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5961 - accuracy:
0.6896 - val_loss: 0.5951 - val_accuracy: 0.6816
Epoch 55/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5970 - accuracy:
0.6878 - val_loss: 0.5958 - val_accuracy: 0.6867
Epoch 56/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5963 - accuracy:
0.6903 - val_loss: 0.6012 - val_accuracy: 0.6898
Epoch 57/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5977 - accuracy:
0.6855 - val_loss: 0.5999 - val_accuracy: 0.6932
Epoch 58/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5964 - accuracy:
```

```
0.6895 - val_loss: 0.5950 - val_accuracy: 0.6850
Epoch 59/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5971 - accuracy:
0.6866 - val_loss: 0.5945 - val_accuracy: 0.6826
Epoch 60/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5959 - accuracy:
0.6888 - val_loss: 0.5990 - val_accuracy: 0.6942
Epoch 61/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5965 - accuracy:
0.6897 - val_loss: 0.5986 - val_accuracy: 0.6745
Epoch 62/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5970 - accuracy:
0.6876 - val_loss: 0.5959 - val_accuracy: 0.6925
Epoch 63/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5954 - accuracy:
0.6916 - val_loss: 0.5945 - val_accuracy: 0.6857
Epoch 64/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5972 - accuracy:
0.6876 - val_loss: 0.5964 - val_accuracy: 0.6775
Epoch 65/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5953 - accuracy:
0.6902 - val_loss: 0.5944 - val_accuracy: 0.6833
Epoch 66/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5949 - accuracy:
0.6917 - val_loss: 0.5953 - val_accuracy: 0.6789
Epoch 67/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5950 - accuracy:
0.6924 - val_loss: 0.5950 - val_accuracy: 0.6789
Epoch 68/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5949 - accuracy:
0.6890 - val_loss: 0.5954 - val_accuracy: 0.6918
Epoch 69/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5951 - accuracy:
0.6932 - val_loss: 0.5954 - val_accuracy: 0.6935
Epoch 70/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5946 - accuracy:
0.6933 - val_loss: 0.5950 - val_accuracy: 0.6786
Epoch 71/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5939 - accuracy:
0.6925 - val_loss: 0.5953 - val_accuracy: 0.6938
Epoch 72/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5938 - accuracy:
0.6946 - val_loss: 0.5938 - val_accuracy: 0.6843
Epoch 73/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5945 - accuracy:
0.6922 - val_loss: 0.5935 - val_accuracy: 0.6830
Epoch 74/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5932 - accuracy:
```

```
0.6955 - val_loss: 0.5949 - val_accuracy: 0.6925
Epoch 75/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5937 - accuracy:
0.6926 - val_loss: 0.5937 - val_accuracy: 0.6840
Epoch 76/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5942 - accuracy:
0.6929 - val_loss: 0.5982 - val_accuracy: 0.6955
Epoch 77/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5947 - accuracy:
0.6891 - val_loss: 0.5950 - val_accuracy: 0.6945
Epoch 78/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5945 - accuracy:
0.6889 - val_loss: 0.5937 - val_accuracy: 0.6874
Epoch 79/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5938 - accuracy:
0.6911 - val_loss: 0.5931 - val_accuracy: 0.6837
Epoch 80/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5935 - accuracy:
0.6935 - val_loss: 0.5930 - val_accuracy: 0.6833
Epoch 81/100
41/41 [==============================] - 0s 4ms/step - loss: 0.5931 - accuracy:
0.6934 - val_loss: 0.5942 - val_accuracy: 0.6908
Epoch 82/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5928 - accuracy:
0.6949 - val_loss: 0.5931 - val_accuracy: 0.6840
Epoch 83/100
41/41 [==============================] - 0s 4ms/step - loss: 0.5940 - accuracy:
0.6904 - val_loss: 0.5945 - val_accuracy: 0.6922
Epoch 84/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5928 - accuracy:
0.6964 - val_loss: 0.5930 - val_accuracy: 0.6884
Epoch 85/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5957 - accuracy:
0.6882 - val_loss: 0.5945 - val_accuracy: 0.6949
Epoch 86/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5937 - accuracy:
0.6925 - val_loss: 0.5931 - val_accuracy: 0.6857
Epoch 87/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5924 - accuracy:
0.6940 - val_loss: 0.5932 - val_accuracy: 0.6888
Epoch 88/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5931 - accuracy:
0.6935 - val_loss: 0.5929 - val_accuracy: 0.6857
Epoch 89/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5936 - accuracy:
0.6940 - val_loss: 0.5935 - val_accuracy: 0.6884
Epoch 90/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5924 - accuracy:
```

```
0.6942 - val_loss: 0.5972 - val_accuracy: 0.6979
Epoch 91/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5935 - accuracy:
0.6943 - val_loss: 0.5958 - val_accuracy: 0.6976
Epoch 92/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5925 - accuracy:
0.6935 - val_loss: 0.5929 - val_accuracy: 0.6847
Epoch 93/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5935 - accuracy:
0.6940 - val_loss: 0.5951 - val_accuracy: 0.6986
Epoch 94/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5920 - accuracy:
0.6949 - val_loss: 0.5962 - val_accuracy: 0.6993
Epoch 95/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5939 - accuracy:
0.6920 - val_loss: 0.5940 - val_accuracy: 0.6952
Epoch 96/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5918 - accuracy:
0.6939 - val_loss: 0.5946 - val_accuracy: 0.6955
Epoch 97/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5924 - accuracy:
0.6925 - val_loss: 0.5982 - val_accuracy: 0.6932
Epoch 98/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5926 - accuracy:
0.6937 - val_loss: 0.5924 - val_accuracy: 0.6857
Epoch 99/100
41/41 [==============================] - 0s 3ms/step - loss: 0.5911 - accuracy:
0.6960 - val_loss: 0.5984 - val_accuracy: 0.6952
Epoch 100/100
41/41 [==============================] - 0s 4ms/step - loss: 0.5920 - accuracy:
0.6960 - val_loss: 0.5922 - val_accuracy: 0.6867
644/644 [==============================] - 1s 719us/step
92/92 [==============================] - 0s 709us/step
82/82 [==============================] - 0s 871us/step
```




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

         0.0      0.692     0.706     0.699     10260
         1.0      0.702     0.688     0.695     10341

    accuracy                          0.697     20601
   macro avg      0.697     0.697     0.697     20601
weighted avg      0.697     0.697     0.697     20601


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.679     0.703     0.691      1466
         1.0      0.694     0.671     0.683      1477

    accuracy                          0.687      2943
   macro avg      0.687     0.687     0.687      2943
weighted avg      0.687     0.687     0.687      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.671     0.687     0.679      1303
         1.0      0.682     0.667     0.674      1314
```

```
    accuracy                           0.677      2617
   macro avg      0.677     0.677      0.677      2617
weighted avg      0.677     0.677      0.677      2617
```

Decision Trees

```
[ ]: from sklearn.tree import DecisionTreeClassifier

dt_cls =  DecisionTreeClassifier(criterion = 'gini', splitter = 'best',␣
 ↪max_depth = 12, min_samples_split = 2, )
dt_cls.fit(x_train.reshape(n_train, -1), y_train)
preds_train = dt_cls.predict(x_train.reshape(n_train, -1))
preds_val   = dt_cls.predict(x_val.reshape(n_val, -1))
preds_test  = dt_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))

# eval = testing_module.ModelEvaluation(y_val,
#     nb_cls.predict(x_val.reshape(n_val, -1)),
#     model_reference_name = 'Naive Bayes',
#     model_type = 'classification',
#     plot_classification_metric = ['roc_auc']) # if classification
# eval.evaluate(evaluate_save= True, plots_show = True)
```

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

         0.0      0.761     0.837     0.797     10260
         1.0      0.821     0.739     0.778     10341

    accuracy                          0.788     20601
   macro avg      0.791     0.788     0.787     20601
weighted avg      0.791     0.788     0.787     20601


Validation Classification Report:
               precision    recall  f1-score   support

         0.0      0.667     0.760     0.711      1466
         1.0      0.724     0.624     0.670      1477

    accuracy                          0.692      2943
```

```
   macro avg      0.696     0.692     0.690      2943
weighted avg      0.696     0.692     0.690      2943


Testing Classification Report:
               precision    recall  f1-score   support

         0.0      0.653     0.740     0.694      1303
         1.0      0.703     0.610     0.653      1314

    accuracy                          0.675      2617
   macro avg      0.678     0.675     0.674      2617
weighted avg      0.678     0.675     0.673      2617
```

XGBoost

[ ]:
```python
tf.keras.backend.clear_session()
from xgboost import XGBClassifier

xgb_cls = XGBClassifier(max_depth = 10, objective = 'reg:logistic',
                           num_parallel_tree = 20, booster = 'gbtree',
                           gamma = 0.5, tree_method = 'gpu_hist', subsample =␣
 ↪0.4, reg_lambda = 1)
setattr(xgb_cls, 'verbosity', 1)
xgb_cls.fit(x_train.reshape(n_train, -1), y_train)
preds_train = xgb_cls.predict(x_train.reshape(n_train, -1))
preds_val   = xgb_cls.predict(x_val.reshape(n_val, -1))
preds_test  = xgb_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

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

         0.0      0.998     0.999     0.999     10260
         1.0      0.999     0.998     0.999     10341

    accuracy                          0.999     20601
   macro avg      0.999     0.999     0.999     20601
weighted avg      0.999     0.999     0.999     20601
```

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

         0.0      0.904     0.954     0.928      1466
         1.0      0.951     0.899     0.924      1477

    accuracy                          0.926      2943
   macro avg      0.927     0.926     0.926      2943
weighted avg      0.928     0.926     0.926      2943


Testing Classification Report:
               precision    recall  f1-score   support

         0.0      0.920     0.959     0.939      1303
         1.0      0.957     0.918     0.937      1314

    accuracy                          0.938      2617
   macro avg      0.939     0.938     0.938      2617
weighted avg      0.939     0.938     0.938      2617
```

SVM

[ ]:
```python
from sklearn.svm import SVC

svm_cls = SVC(kernel = 'poly', degree = 3, gamma = 'auto', max_iter = 250,␣
 ↪verbose= True)
svm_cls.fit(x_train.reshape(n_train, -1), y_train)
preds_train = svm_cls.predict(x_train.reshape(n_train, -1))
preds_val   = svm_cls.predict(x_val.reshape(n_val, -1))
preds_test  = svm_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
[LibSVM]
```

```
c:\ProgramData\Anaconda3\envs\ml\lib\site-packages\sklearn\svm\_base.py:301:
ConvergenceWarning: Solver terminated early (max_iter=250).  Consider pre-
processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(
```

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

```
         0.0      0.200     0.000     0.000     10260
         1.0      0.502     1.000     0.668     10341

    accuracy                          0.502     20601
   macro avg      0.351     0.500     0.334     20601
weighted avg      0.352     0.502     0.336     20601


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1466
         1.0      0.502     1.000     0.668      1477

    accuracy                          0.502      2943
   macro avg      0.251     0.500     0.334      2943
weighted avg      0.252     0.502     0.335      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.000     0.000     0.000      1303
         1.0      0.502     0.999     0.668      1314

    accuracy                          0.502      2617
   macro avg      0.251     0.500     0.334      2617
weighted avg      0.252     0.502     0.336      2617

c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
c:\ProgramData\Anaconda3\envs\ml\lib\site-
packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning:
Precision and F-score are 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, msg_start, len(result))
```

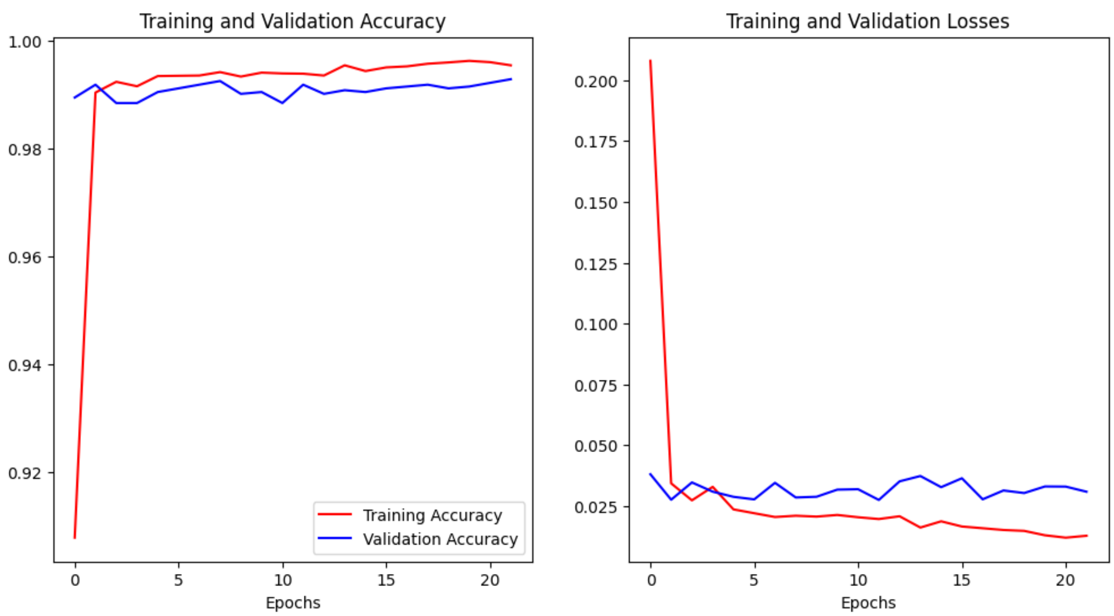
CNN Model

```
[ ]: cnn = tf.keras.Sequential([
        tf.keras.layers.Conv2D(64, (3,3), padding = 'same', activation = 'relu',␣
     ↪input_shape = x_train[0].shape),
        tf.keras.layers.MaxPool2D((3,3), padding = 'same'),

        tf.keras.layers.UpSampling2D((4,4)),

        tf.keras.layers.Conv2D(32, (3,3), padding = 'same', activation = 'relu'),
        tf.keras.layers.MaxPool2D((3,3), padding = 'same'),

        tf.keras.layers.Conv2D(32, (2,2), padding = 'same', activation = 'relu'),
        tf.keras.layers.MaxPool2D((2,2), padding = 'same'),


        tf.keras.layers.Flatten(),

        tf.keras.layers.Dense(512, activation = 'relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(512, activation = 'relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(2, activation = 'sigmoid')
    ])

    cnn.compile(
        optimizer = tf.keras.optimizers.Adam(learning_rate=0.003),
        loss = 'sparse_categorical_crossentropy',
        metrics = ['accuracy']
    )

    history = cnn.fit(
                x_train, y_train, batch_size = 256,
                epochs = 50,
                validation_data = [x_val, y_val],
                validation_batch_size = 256,
                callbacks = [tf.keras.callbacks.EarlyStopping(monitor = 'val_loss',␣
     ↪patience = 10),
                             tf.keras.callbacks.ModelCheckpoint(str = 'val_loss',␣
     ↪save_best_only = True, save_weights_only = True, filepath= "CNN_No_Aug_HSV")]
            )


    cnn.load_weights('CNN_No_Aug_HSV')
    fig, ax = plot_history(history)

    fig.savefig(PLOT_SAVE + "CNN_unaug_losscurve_HSV.png", facecolor = 'white')
    preds_train = np.argmax(cnn.predict(x_train), axis = 1)
    preds_val   = np.argmax(cnn.predict(x_val), axis = 1)
```

```python
preds_test  = np.argmax(cnn.predict(x_test), axis = 1)

print("Training Classification Report: \n", classification_report(y_train,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/50
81/81 [==============================] - 3s 17ms/step - loss: 0.2080 - accuracy:
0.9079 - val_loss: 0.0381 - val_accuracy: 0.9895
Epoch 2/50
81/81 [==============================] - 1s 12ms/step - loss: 0.0343 - accuracy:
0.9904 - val_loss: 0.0276 - val_accuracy: 0.9918
Epoch 3/50
81/81 [==============================] - 1s 12ms/step - loss: 0.0274 - accuracy:
0.9924 - val_loss: 0.0347 - val_accuracy: 0.9884
Epoch 4/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0329 - accuracy:
0.9916 - val_loss: 0.0309 - val_accuracy: 0.9884
Epoch 5/50
81/81 [==============================] - 1s 12ms/step - loss: 0.0237 - accuracy:
0.9934 - val_loss: 0.0288 - val_accuracy: 0.9905
Epoch 6/50
81/81 [==============================] - 1s 12ms/step - loss: 0.0221 - accuracy:
0.9935 - val_loss: 0.0278 - val_accuracy: 0.9912
Epoch 7/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0205 - accuracy:
0.9935 - val_loss: 0.0346 - val_accuracy: 0.9918
Epoch 8/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0210 - accuracy:
0.9942 - val_loss: 0.0285 - val_accuracy: 0.9925
Epoch 9/50
81/81 [==============================] - 1s 12ms/step - loss: 0.0207 - accuracy:
0.9933 - val_loss: 0.0288 - val_accuracy: 0.9901
Epoch 10/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0214 - accuracy:
0.9941 - val_loss: 0.0318 - val_accuracy: 0.9905
Epoch 11/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0204 - accuracy:
0.9939 - val_loss: 0.0319 - val_accuracy: 0.9884
Epoch 12/50
81/81 [==============================] - 1s 12ms/step - loss: 0.0197 - accuracy:
0.9939 - val_loss: 0.0275 - val_accuracy: 0.9918
Epoch 13/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0208 - accuracy:
```

```
0.9935 - val_loss: 0.0352 - val_accuracy: 0.9901
Epoch 14/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0162 - accuracy:
0.9954 - val_loss: 0.0374 - val_accuracy: 0.9908
Epoch 15/50
81/81 [==============================] - 1s 12ms/step - loss: 0.0187 - accuracy:
0.9944 - val_loss: 0.0328 - val_accuracy: 0.9905
Epoch 16/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0166 - accuracy:
0.9950 - val_loss: 0.0364 - val_accuracy: 0.9912
Epoch 17/50
81/81 [==============================] - 1s 12ms/step - loss: 0.0159 - accuracy:
0.9952 - val_loss: 0.0278 - val_accuracy: 0.9915
Epoch 18/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0152 - accuracy:
0.9957 - val_loss: 0.0314 - val_accuracy: 0.9918
Epoch 19/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0148 - accuracy:
0.9960 - val_loss: 0.0304 - val_accuracy: 0.9912
Epoch 20/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0130 - accuracy:
0.9963 - val_loss: 0.0331 - val_accuracy: 0.9915
Epoch 21/50
81/81 [==============================] - 1s 12ms/step - loss: 0.0120 - accuracy:
0.9960 - val_loss: 0.0330 - val_accuracy: 0.9922
Epoch 22/50
81/81 [==============================] - 1s 11ms/step - loss: 0.0128 - accuracy:
0.9954 - val_loss: 0.0309 - val_accuracy: 0.9929
```



```
644/644 [==============================] - 1s 1ms/step
92/92 [==============================] - 0s 2ms/step
82/82 [==============================] - 0s 1ms/step
Training Classification Report:
              precision    recall  f1-score   support

         0.0      0.996     0.995     0.996     10260
         1.0      0.995     0.996     0.996     10341

    accuracy                          0.996     20601
   macro avg      0.996     0.996     0.996     20601
weighted avg      0.996     0.996     0.996     20601


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.993     0.990     0.992      1466
         1.0      0.991     0.993     0.992      1477

    accuracy                          0.992      2943
   macro avg      0.992     0.992     0.992      2943
weighted avg      0.992     0.992     0.992      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.994     0.992     0.993      1303
         1.0      0.992     0.994     0.993      1314

    accuracy                          0.993      2617
   macro avg      0.993     0.993     0.993      2617
weighted avg      0.993     0.993     0.993      2617
```

0.7.3 Augmented Dataset

Naive Bayes

[ ]:
```
from sklearn import naive_bayes

nb_cls = naive_bayes.GaussianNB()
nb_cls.fit(x_train_aug.reshape(n_aug_train, -1), y_train_aug)
preds_train = nb_cls.predict(x_train_aug.reshape(n_aug_train, -1))
preds_val   = nb_cls.predict(x_val.reshape(n_val, -1))
preds_test  = nb_cls.predict(x_test.reshape(n_test, -1))
```

```
print("Training Classification Report: \n", classification_report(y_train_aug,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

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

         0.0      0.607     0.612     0.609     20520
         1.0      0.612     0.607     0.609     20682

    accuracy                          0.609     41202
   macro avg      0.609     0.609     0.609     41202
weighted avg      0.609     0.609     0.609     41202


Validation Classification Report:
              precision    recall  f1-score   support

         0.0      0.610     0.493     0.545      1466
         1.0      0.577     0.687     0.627      1477

    accuracy                          0.590      2943
   macro avg      0.593     0.590     0.586      2943
weighted avg      0.593     0.590     0.586      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.653     0.495     0.563      1303
         1.0      0.596     0.739     0.660      1314

    accuracy                          0.618      2617
   macro avg      0.624     0.617     0.611      2617
weighted avg      0.624     0.618     0.612      2617
```

Logistic Regression

[ ]:
```
from sklearn.linear_model import LogisticRegression

logreg_cls =  tf.keras.Sequential([tf.keras.layers.Dense(2, activation =␣
 ↪'sigmoid', input_dim = HEIGHT * WIDTH)])
```

```
logreg_cls.compile(optimizer = tf.optimizers.Adam(0.0001), loss =␣
↪'sparse_categorical_crossentropy', metrics = ['accuracy'])
history = logreg_cls.fit(x_train_aug.reshape(n_aug_train, -1),
                         y_train_aug,
                         batch_size = 512,
                         validation_data=[x_val.reshape(n_val, -1), y_val],
                         validation_batch_size=128,
                         epochs = 100,
                         callbacks =  [tf.keras.callbacks.EarlyStopping(monitor␣
↪= 'val_loss', patience = 10),
                                       tf.keras.callbacks.ModelCheckpoint(str =␣
↪'val_loss', save_best_only = True, save_weights_only = True, filepath=␣
↪"LR_Aug_HSV")])
preds_train = np.argmax(logreg_cls.predict(x_train_aug.reshape(n_aug_train,␣
↪-1)), axis = 1)
preds_val   = np.argmax(logreg_cls.predict(x_val.reshape(n_val, -1)), axis = 1)
preds_test  = np.argmax(logreg_cls.predict(x_test.reshape(n_test, -1)), axis =␣
↪1)

logreg_cls.load_weights('LR_Aug_HSV')
fig, ax = plot_history(history)

fig.savefig(PLOT_SAVE + "logreg_aug_losscurve_HSV.png", facecolor = 'white')

print("Training Classification Report: \n", classification_report(y_train_aug,␣
↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6949 - accuracy:
0.4968 - val_loss: 0.6917 - val_accuracy: 0.4910
Epoch 2/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6887 - accuracy:
0.4986 - val_loss: 0.6852 - val_accuracy: 0.4937
Epoch 3/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6855 - accuracy:
0.5055 - val_loss: 0.6825 - val_accuracy: 0.4975
Epoch 4/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6822 - accuracy:
0.5168 - val_loss: 0.6801 - val_accuracy: 0.5158
Epoch 5/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6792 - accuracy:
0.5251 - val_loss: 0.6773 - val_accuracy: 0.5304
Epoch 6/100
```

```
81/81 [==============================] - 0s 2ms/step - loss: 0.6763 - accuracy:
0.5403 - val_loss: 0.6743 - val_accuracy: 0.5375
Epoch 7/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6736 - accuracy:
0.5521 - val_loss: 0.6727 - val_accuracy: 0.5627
Epoch 8/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6713 - accuracy:
0.5638 - val_loss: 0.6701 - val_accuracy: 0.5657
Epoch 9/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6690 - accuracy:
0.5757 - val_loss: 0.6682 - val_accuracy: 0.5691
Epoch 10/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6668 - accuracy:
0.5844 - val_loss: 0.6665 - val_accuracy: 0.5797
Epoch 11/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6649 - accuracy:
0.5925 - val_loss: 0.6648 - val_accuracy: 0.5841
Epoch 12/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6631 - accuracy:
0.6009 - val_loss: 0.6634 - val_accuracy: 0.6018
Epoch 13/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6614 - accuracy:
0.6089 - val_loss: 0.6623 - val_accuracy: 0.5841
Epoch 14/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6597 - accuracy:
0.6125 - val_loss: 0.6605 - val_accuracy: 0.6082
Epoch 15/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6582 - accuracy:
0.6184 - val_loss: 0.6594 - val_accuracy: 0.6086
Epoch 16/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6568 - accuracy:
0.6232 - val_loss: 0.6580 - val_accuracy: 0.6140
Epoch 17/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6555 - accuracy:
0.6254 - val_loss: 0.6568 - val_accuracy: 0.6140
Epoch 18/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6543 - accuracy:
0.6281 - val_loss: 0.6560 - val_accuracy: 0.6140
Epoch 19/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6531 - accuracy:
0.6299 - val_loss: 0.6547 - val_accuracy: 0.6171
Epoch 20/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6519 - accuracy:
0.6340 - val_loss: 0.6548 - val_accuracy: 0.6130
Epoch 21/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6509 - accuracy:
0.6341 - val_loss: 0.6533 - val_accuracy: 0.6154
Epoch 22/100
```

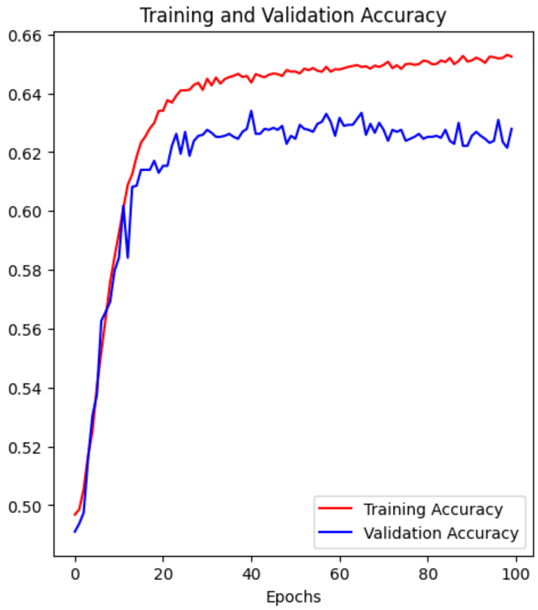
```
81/81 [==============================] - 0s 2ms/step - loss: 0.6498 - accuracy:
0.6377 - val_loss: 0.6530 - val_accuracy: 0.6154
Epoch 23/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6489 - accuracy:
0.6368 - val_loss: 0.6512 - val_accuracy: 0.6222
Epoch 24/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6479 - accuracy:
0.6392 - val_loss: 0.6500 - val_accuracy: 0.6262
Epoch 25/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6471 - accuracy:
0.6410 - val_loss: 0.6504 - val_accuracy: 0.6194
Epoch 26/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6463 - accuracy:
0.6410 - val_loss: 0.6485 - val_accuracy: 0.6269
Epoch 27/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6454 - accuracy:
0.6412 - val_loss: 0.6498 - val_accuracy: 0.6188
Epoch 28/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6447 - accuracy:
0.6429 - val_loss: 0.6478 - val_accuracy: 0.6239
Epoch 29/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6439 - accuracy:
0.6436 - val_loss: 0.6466 - val_accuracy: 0.6256
Epoch 30/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6432 - accuracy:
0.6412 - val_loss: 0.6460 - val_accuracy: 0.6259
Epoch 31/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6425 - accuracy:
0.6449 - val_loss: 0.6458 - val_accuracy: 0.6276
Epoch 32/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6418 - accuracy:
0.6427 - val_loss: 0.6444 - val_accuracy: 0.6266
Epoch 33/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6411 - accuracy:
0.6454 - val_loss: 0.6454 - val_accuracy: 0.6252
Epoch 34/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6407 - accuracy:
0.6433 - val_loss: 0.6440 - val_accuracy: 0.6252
Epoch 35/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6399 - accuracy:
0.6448 - val_loss: 0.6432 - val_accuracy: 0.6256
Epoch 36/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6394 - accuracy:
0.6455 - val_loss: 0.6426 - val_accuracy: 0.6262
Epoch 37/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6388 - accuracy:
0.6459 - val_loss: 0.6432 - val_accuracy: 0.6252
Epoch 38/100
```

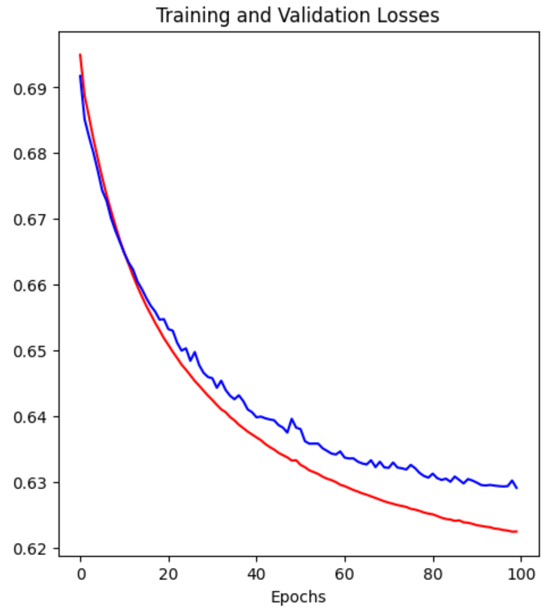
```
81/81 [==============================] - 0s 3ms/step - loss: 0.6382 - accuracy:
0.6466 - val_loss: 0.6424 - val_accuracy: 0.6245
Epoch 39/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6377 - accuracy:
0.6456 - val_loss: 0.6411 - val_accuracy: 0.6269
Epoch 40/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6373 - accuracy:
0.6459 - val_loss: 0.6406 - val_accuracy: 0.6279
Epoch 41/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6368 - accuracy:
0.6437 - val_loss: 0.6399 - val_accuracy: 0.6340
Epoch 42/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6364 - accuracy:
0.6465 - val_loss: 0.6400 - val_accuracy: 0.6262
Epoch 43/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6358 - accuracy:
0.6459 - val_loss: 0.6397 - val_accuracy: 0.6262
Epoch 44/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6354 - accuracy:
0.6455 - val_loss: 0.6396 - val_accuracy: 0.6279
Epoch 45/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6350 - accuracy:
0.6463 - val_loss: 0.6394 - val_accuracy: 0.6276
Epoch 46/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6345 - accuracy:
0.6467 - val_loss: 0.6387 - val_accuracy: 0.6283
Epoch 47/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6341 - accuracy:
0.6465 - val_loss: 0.6383 - val_accuracy: 0.6276
Epoch 48/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6338 - accuracy:
0.6459 - val_loss: 0.6376 - val_accuracy: 0.6290
Epoch 49/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6333 - accuracy:
0.6479 - val_loss: 0.6397 - val_accuracy: 0.6228
Epoch 50/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6334 - accuracy:
0.6474 - val_loss: 0.6383 - val_accuracy: 0.6256
Epoch 51/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6327 - accuracy:
0.6474 - val_loss: 0.6381 - val_accuracy: 0.6245
Epoch 52/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6323 - accuracy:
0.6468 - val_loss: 0.6363 - val_accuracy: 0.6293
Epoch 53/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6318 - accuracy:
0.6484 - val_loss: 0.6359 - val_accuracy: 0.6279
Epoch 54/100
```

```
81/81 [==============================] - 0s 3ms/step - loss: 0.6316 - accuracy:
0.6479 - val_loss: 0.6359 - val_accuracy: 0.6276
Epoch 55/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6313 - accuracy:
0.6486 - val_loss: 0.6359 - val_accuracy: 0.6269
Epoch 56/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6309 - accuracy:
0.6477 - val_loss: 0.6352 - val_accuracy: 0.6296
Epoch 57/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6306 - accuracy:
0.6473 - val_loss: 0.6348 - val_accuracy: 0.6303
Epoch 58/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6303 - accuracy:
0.6490 - val_loss: 0.6344 - val_accuracy: 0.6330
Epoch 59/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6300 - accuracy:
0.6473 - val_loss: 0.6342 - val_accuracy: 0.6303
Epoch 60/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6296 - accuracy:
0.6482 - val_loss: 0.6347 - val_accuracy: 0.6256
Epoch 61/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6294 - accuracy:
0.6481 - val_loss: 0.6338 - val_accuracy: 0.6317
Epoch 62/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6291 - accuracy:
0.6485 - val_loss: 0.6336 - val_accuracy: 0.6290
Epoch 63/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6288 - accuracy:
0.6490 - val_loss: 0.6336 - val_accuracy: 0.6293
Epoch 64/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6286 - accuracy:
0.6493 - val_loss: 0.6331 - val_accuracy: 0.6293
Epoch 65/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6283 - accuracy:
0.6496 - val_loss: 0.6329 - val_accuracy: 0.6313
Epoch 66/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6281 - accuracy:
0.6489 - val_loss: 0.6327 - val_accuracy: 0.6334
Epoch 67/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6279 - accuracy:
0.6492 - val_loss: 0.6334 - val_accuracy: 0.6259
Epoch 68/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6276 - accuracy:
0.6484 - val_loss: 0.6323 - val_accuracy: 0.6296
Epoch 69/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6274 - accuracy:
0.6494 - val_loss: 0.6331 - val_accuracy: 0.6266
Epoch 70/100
```

```
81/81 [==============================] - 0s 2ms/step - loss: 0.6271 - accuracy:
0.6489 - val_loss: 0.6323 - val_accuracy: 0.6300
Epoch 71/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6269 - accuracy:
0.6497 - val_loss: 0.6322 - val_accuracy: 0.6276
Epoch 72/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6267 - accuracy:
0.6507 - val_loss: 0.6330 - val_accuracy: 0.6239
Epoch 73/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6265 - accuracy:
0.6486 - val_loss: 0.6322 - val_accuracy: 0.6276
Epoch 74/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6264 - accuracy:
0.6496 - val_loss: 0.6321 - val_accuracy: 0.6269
Epoch 75/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6263 - accuracy:
0.6483 - val_loss: 0.6319 - val_accuracy: 0.6276
Epoch 76/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6260 - accuracy:
0.6499 - val_loss: 0.6326 - val_accuracy: 0.6239
Epoch 77/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6259 - accuracy:
0.6501 - val_loss: 0.6321 - val_accuracy: 0.6245
Epoch 78/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6257 - accuracy:
0.6497 - val_loss: 0.6314 - val_accuracy: 0.6252
Epoch 79/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6254 - accuracy:
0.6499 - val_loss: 0.6310 - val_accuracy: 0.6262
Epoch 80/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6253 - accuracy:
0.6511 - val_loss: 0.6307 - val_accuracy: 0.6245
Epoch 81/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6251 - accuracy:
0.6508 - val_loss: 0.6313 - val_accuracy: 0.6252
Epoch 82/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6249 - accuracy:
0.6499 - val_loss: 0.6306 - val_accuracy: 0.6252
Epoch 83/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6246 - accuracy:
0.6500 - val_loss: 0.6304 - val_accuracy: 0.6256
Epoch 84/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6244 - accuracy:
0.6512 - val_loss: 0.6306 - val_accuracy: 0.6249
Epoch 85/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6243 - accuracy:
0.6506 - val_loss: 0.6301 - val_accuracy: 0.6276
Epoch 86/100
```

```
81/81 [==============================] - 0s 2ms/step - loss: 0.6241 - accuracy:
0.6520 - val_loss: 0.6309 - val_accuracy: 0.6239
Epoch 87/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6242 - accuracy:
0.6499 - val_loss: 0.6304 - val_accuracy: 0.6228
Epoch 88/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6239 - accuracy:
0.6510 - val_loss: 0.6298 - val_accuracy: 0.6300
Epoch 89/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6239 - accuracy:
0.6527 - val_loss: 0.6305 - val_accuracy: 0.6222
Epoch 90/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6237 - accuracy:
0.6508 - val_loss: 0.6303 - val_accuracy: 0.6222
Epoch 91/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6235 - accuracy:
0.6512 - val_loss: 0.6300 - val_accuracy: 0.6256
Epoch 92/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6234 - accuracy:
0.6521 - val_loss: 0.6296 - val_accuracy: 0.6269
Epoch 93/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6233 - accuracy:
0.6514 - val_loss: 0.6295 - val_accuracy: 0.6256
Epoch 94/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6232 - accuracy:
0.6504 - val_loss: 0.6296 - val_accuracy: 0.6245
Epoch 95/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6230 - accuracy:
0.6524 - val_loss: 0.6295 - val_accuracy: 0.6232
Epoch 96/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6229 - accuracy:
0.6523 - val_loss: 0.6294 - val_accuracy: 0.6239
Epoch 97/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6228 - accuracy:
0.6518 - val_loss: 0.6294 - val_accuracy: 0.6310
Epoch 98/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6227 - accuracy:
0.6520 - val_loss: 0.6294 - val_accuracy: 0.6235
Epoch 99/100
81/81 [==============================] - 0s 3ms/step - loss: 0.6225 - accuracy:
0.6531 - val_loss: 0.6303 - val_accuracy: 0.6215
Epoch 100/100
81/81 [==============================] - 0s 2ms/step - loss: 0.6225 - accuracy:
0.6525 - val_loss: 0.6292 - val_accuracy: 0.6279
1288/1288 [==============================] - 1s 685us/step
92/92 [==============================] - 0s 758us/step
82/82 [==============================] - 0s 834us/step
```





Training Classification Report:

	precision	recall	f1-score	support
0.0	0.653	0.654	0.654	20520
1.0	0.656	0.656	0.656	20682
accuracy			0.655	41202
macro avg	0.655	0.655	0.655	41202
weighted avg	0.655	0.655	0.655	41202

Validation Classification Report:

	precision	recall	f1-score	support
0.0	0.639	0.583	0.610	1466
1.0	0.619	0.672	0.645	1477
accuracy			0.628	2943
macro avg	0.629	0.628	0.627	2943
weighted avg	0.629	0.628	0.627	2943

Testing Classification Report:

	precision	recall	f1-score	support
0.0	0.641	0.560	0.598	1303
1.0	0.613	0.689	0.649	1314

```
    accuracy                          0.625      2617
   macro avg      0.627     0.625     0.623      2617
weighted avg      0.627     0.625     0.624      2617
```

Decision Trees

```
[ ]: from sklearn.tree import DecisionTreeClassifier

dt_cls =  DecisionTreeClassifier(criterion = 'gini', splitter = 'best',␣
 ↪max_depth = 25, min_samples_split = 2, )
dt_cls.fit(x_train_aug.reshape(n_aug_train, -1), y_train_aug)
preds_train = dt_cls.predict(x_train_aug.reshape(n_aug_train, -1))
preds_val   = dt_cls.predict(x_val.reshape(n_val, -1))
preds_test  = dt_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train_aug,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

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

         0.0      0.799     0.998     0.887     20520
         1.0      0.997     0.751     0.856     20682

    accuracy                          0.874     41202
   macro avg      0.898     0.874     0.872     41202
weighted avg      0.898     0.874     0.872     41202


Validation Classification Report:
               precision    recall  f1-score   support

         0.0      0.709     0.947     0.811      1466
         1.0      0.921     0.614     0.737      1477

    accuracy                          0.780      2943
   macro avg      0.815     0.780     0.774      2943
weighted avg      0.815     0.780     0.774      2943


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

```
         0.0      0.719     0.944     0.816      1303
         1.0      0.920     0.635     0.751      1314

    accuracy                          0.789      2617
   macro avg      0.819     0.789     0.784      2617
weighted avg      0.820     0.789     0.784      2617
```

XGBoost

```
[ ]: tf.keras.backend.clear_session()

from xgboost import XGBClassifier

xgb_cls = XGBClassifier(max_depth = 10, objective = 'reg:logistic',
                            num_parallel_tree = 20, booster = 'gbtree',
                            gamma = 0.5, tree_method = 'gpu_hist', subsample =␣
 ↪0.4, reg_lambda = 1)
xgb_cls.fit(x_train_aug.reshape(n_aug_train, -1), y_train_aug)
preds_train = xgb_cls.predict(x_train_aug.reshape(n_aug_train, -1))
preds_val   = xgb_cls.predict(x_val.reshape(n_val, -1))
preds_test  = xgb_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train_aug,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

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

         0.0      0.986     0.999     0.993     20520
         1.0      0.999     0.986     0.992     20682

    accuracy                          0.993     41202
   macro avg      0.993     0.993     0.993     41202
weighted avg      0.993     0.993     0.993     41202


Validation Classification Report:
               precision    recall  f1-score   support

         0.0      0.899     0.951     0.924      1466
         1.0      0.948     0.894     0.921      1477

    accuracy                          0.923      2943
```

```
   macro avg      0.924     0.923     0.922      2943
weighted avg      0.924     0.923     0.922      2943


Testing Classification Report:
               precision    recall  f1-score   support

         0.0      0.909     0.954     0.931      1303
         1.0      0.952     0.906     0.928      1314

    accuracy                          0.930      2617
   macro avg      0.931     0.930     0.930      2617
weighted avg      0.931     0.930     0.930      2617
```

SVM

[ ]:
```
from sklearn.svm import SVC

svm_cls = SVC(kernel = 'rbf', max_iter = 250, verbose= True)
svm_cls.fit(x_train_aug.reshape(n_aug_train, -1), y_train_aug)
preds_train = svm_cls.predict(x_train_aug.reshape(n_aug_train, -1))
preds_val   = svm_cls.predict(x_val.reshape(n_val, -1))
preds_test  = svm_cls.predict(x_test.reshape(n_test, -1))

print("Training Classification Report: \n", classification_report(y_train_aug,␣
  ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
  ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
  ↪preds_test, digits = N_DIGITS))
```

```
[LibSVM]

c:\ProgramData\Anaconda3\envs\ml\lib\site-packages\sklearn\svm\_base.py:301:
ConvergenceWarning: Solver terminated early (max_iter=250).  Consider pre-
processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(

Training Classification Report:
               precision    recall  f1-score   support

         0.0      0.437     0.618     0.512     20520
         1.0      0.357     0.211     0.265     20682

    accuracy                          0.413     41202
   macro avg      0.397     0.414     0.388     41202
weighted avg      0.397     0.413     0.388     41202
```

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

         0.0      0.432     0.596     0.501      1466
         1.0      0.357     0.222     0.274      1477

    accuracy                          0.408      2943
   macro avg      0.394     0.409     0.387      2943
weighted avg      0.394     0.408     0.387      2943


Testing Classification Report:
              precision    recall  f1-score   support

         0.0      0.423     0.601     0.497      1303
         1.0      0.323     0.189     0.238      1314

    accuracy                          0.394      2617
   macro avg      0.373     0.395     0.368      2617
weighted avg      0.373     0.394     0.367      2617
```

CNN Model

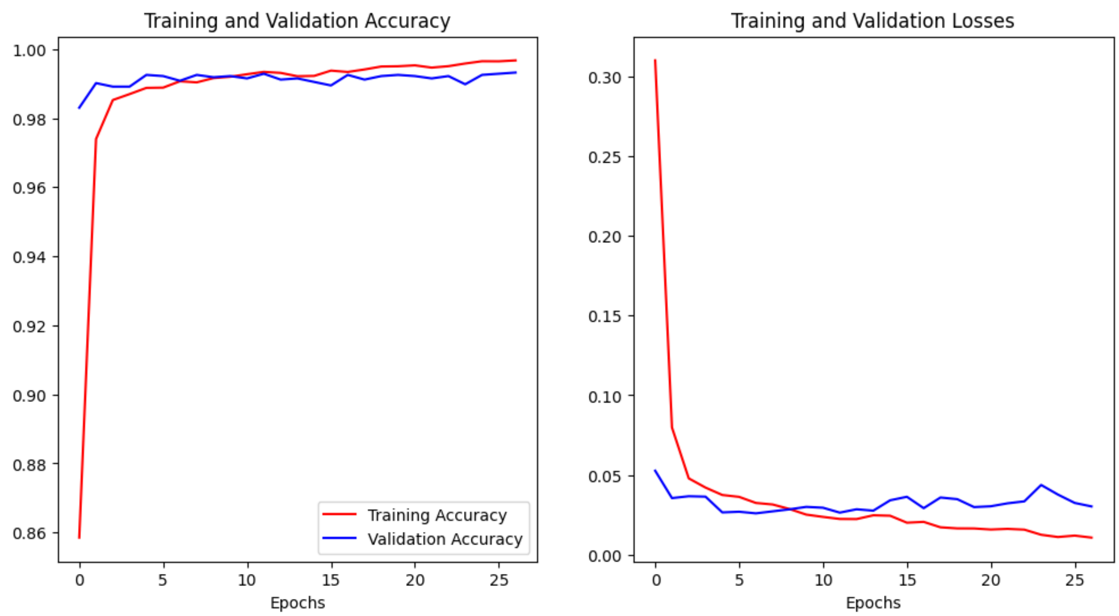
[ ]:
```
cnn = tf.keras.Sequential([
    tf.keras.layers.Conv2D(64, (3,3), padding = 'same', activation = 'relu',␣
 ↪input_shape = x_train[0].shape),
    tf.keras.layers.MaxPool2D((3,3), padding = 'same'),

    tf.keras.layers.UpSampling2D((4,4)),

    tf.keras.layers.Conv2D(32, (3,3), padding = 'same', activation = 'relu'),
    tf.keras.layers.MaxPool2D((3,3), padding = 'same'),

    tf.keras.layers.Conv2D(32, (2,2), padding = 'same', activation = 'relu'),
    tf.keras.layers.MaxPool2D((2,2), padding = 'same'),


    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(2, activation = 'sigmoid')
])
```

```python
cnn.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.001),
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)

history = cnn.fit(
            x_train_aug, y_train_aug, batch_size = 512,
            epochs = 100,
            shuffle = True,
            validation_data = [x_val, y_val],
            validation_batch_size = 256,
            callbacks = [
                tf.keras.callbacks.EarlyStopping(monitor = 'val_loss', patience␣
 ↪= 20),
                tf.keras.callbacks.ModelCheckpoint(str = 'val_loss',␣
 ↪save_best_only = True, save_weights_only = True, filepath= "CNN_Aug_HSV")
            ]
        )

fig, ax = plot_history(history)

cnn.load_weights('CNN_Aug_HSV')

fig.savefig(PLOT_SAVE + "CNN_aug_losscurve_HSV.png", facecolor = 'white')
preds_train = np.argmax(cnn.predict(x_train_aug), axis = 1)
preds_val   = np.argmax(cnn.predict(x_val), axis = 1)
preds_test  = np.argmax(cnn.predict(x_test), axis = 1)

print("Training Classification Report: \n", classification_report(y_train_aug,␣
 ↪preds_train, digits = N_DIGITS))
print("\nValidation Classification Report: \n", classification_report(y_val,␣
 ↪preds_val, digits = N_DIGITS))
print("\nTesting Classification Report: \n", classification_report(y_test,␣
 ↪preds_test, digits = N_DIGITS))
```

```
Epoch 1/100
81/81 [==============================] - 5s 28ms/step - loss: 0.3102 - accuracy:
0.8585 - val_loss: 0.0527 - val_accuracy: 0.9830
Epoch 2/100
81/81 [==============================] - 2s 22ms/step - loss: 0.0798 - accuracy:
0.9740 - val_loss: 0.0355 - val_accuracy: 0.9901
Epoch 3/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0479 - accuracy:
0.9852 - val_loss: 0.0367 - val_accuracy: 0.9891
Epoch 4/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0421 - accuracy:
```

```
0.9870 - val_loss: 0.0364 - val_accuracy: 0.9891
Epoch 5/100
81/81 [==============================] - 2s 22ms/step - loss: 0.0374 - accuracy:
0.9888 - val_loss: 0.0265 - val_accuracy: 0.9925
Epoch 6/100
81/81 [==============================] - 2s 22ms/step - loss: 0.0363 - accuracy:
0.9888 - val_loss: 0.0269 - val_accuracy: 0.9922
Epoch 7/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0325 - accuracy:
0.9907 - val_loss: 0.0259 - val_accuracy: 0.9908
Epoch 8/100
81/81 [==============================] - 2s 20ms/step - loss: 0.0315 - accuracy:
0.9904 - val_loss: 0.0272 - val_accuracy: 0.9925
Epoch 9/100
81/81 [==============================] - 2s 19ms/step - loss: 0.0287 - accuracy:
0.9916 - val_loss: 0.0285 - val_accuracy: 0.9918
Epoch 10/100
81/81 [==============================] - 2s 19ms/step - loss: 0.0251 - accuracy:
0.9920 - val_loss: 0.0300 - val_accuracy: 0.9922
Epoch 11/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0237 - accuracy:
0.9927 - val_loss: 0.0295 - val_accuracy: 0.9915
Epoch 12/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0224 - accuracy:
0.9934 - val_loss: 0.0264 - val_accuracy: 0.9929
Epoch 13/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0223 - accuracy:
0.9931 - val_loss: 0.0285 - val_accuracy: 0.9912
Epoch 14/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0247 - accuracy:
0.9921 - val_loss: 0.0276 - val_accuracy: 0.9915
Epoch 15/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0244 - accuracy:
0.9922 - val_loss: 0.0341 - val_accuracy: 0.9905
Epoch 16/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0201 - accuracy:
0.9938 - val_loss: 0.0363 - val_accuracy: 0.9895
Epoch 17/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0206 - accuracy:
0.9934 - val_loss: 0.0292 - val_accuracy: 0.9925
Epoch 18/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0172 - accuracy:
0.9941 - val_loss: 0.0359 - val_accuracy: 0.9912
Epoch 19/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0165 - accuracy:
0.9950 - val_loss: 0.0348 - val_accuracy: 0.9922
Epoch 20/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0164 - accuracy:
```

```
0.9950 - val_loss: 0.0299 - val_accuracy: 0.9925
Epoch 21/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0158 - accuracy:
0.9953 - val_loss: 0.0304 - val_accuracy: 0.9922
Epoch 22/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0162 - accuracy:
0.9946 - val_loss: 0.0322 - val_accuracy: 0.9915
Epoch 23/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0157 - accuracy:
0.9950 - val_loss: 0.0335 - val_accuracy: 0.9922
Epoch 24/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0125 - accuracy:
0.9958 - val_loss: 0.0437 - val_accuracy: 0.9898
Epoch 25/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0111 - accuracy:
0.9965 - val_loss: 0.0377 - val_accuracy: 0.9925
Epoch 26/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0120 - accuracy:
0.9965 - val_loss: 0.0325 - val_accuracy: 0.9929
Epoch 27/100
81/81 [==============================] - 2s 21ms/step - loss: 0.0107 - accuracy:
0.9967 - val_loss: 0.0303 - val_accuracy: 0.9932
```




```
1288/1288 [==============================] - 2s 1ms/step
92/92 [==============================] - 0s 2ms/step
82/82 [==============================] - 0s 2ms/step
Training Classification Report:
```

	precision	recall	f1-score	support
0.0	0.986	0.994	0.990	20520
1.0	0.994	0.986	0.990	20682
accuracy			0.990	41202
macro avg	0.990	0.990	0.990	41202
weighted avg	0.990	0.990	0.990	41202

Validation Classification Report:

	precision	recall	f1-score	support
0.0	0.996	0.986	0.991	1466
1.0	0.986	0.996	0.991	1477
accuracy			0.991	2943
macro avg	0.991	0.991	0.991	2943
weighted avg	0.991	0.991	0.991	2943

Testing Classification Report:

	precision	recall	f1-score	support
0.0	0.996	0.992	0.994	1303
1.0	0.992	0.996	0.994	1314
accuracy			0.994	2617
macro avg	0.994	0.994	0.994	2617
weighted avg	0.994	0.994	0.994	2617