In [37]:
```
import pyspark
from pyspark.sql import SparkSession, SQLContext
from pyspark.ml import Pipeline,Transformer
from pyspark.ml.feature import Imputer,StandardScaler,StringIndexer,OneHotEncoder, Vecto

from pyspark.sql.functions import *
from pyspark.sql.types import *
import numpy as np

col_names = ["duration","protocol_type","service","flag","src_bytes",
"dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
"logged_in","num_compromised","root_shell","su_attempted","num_root",
"num_file_creations","num_shells","num_access_files","num_outbound_cmds",
"is_host_login","is_guest_login","count","srv_count","serror_rate",
"srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
"diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
"dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
"dst_host_rerror_rate","dst_host_srv_rerror_rate","class","difficulty"]

attack_category = ['normal', 'DOS', 'R2L', 'U2R', 'probe']
normal = ['normal']
DOS = ['apache2','back','land','neptune','mailbomb','pod','processtable','smurf','teardr
R2L = ['ftp_write','guess_passwd','httptunnel','imap','multihop','named','phf','sendmail
U2R = ['buffer_overflow','loadmodule','perl','ps','rootkit','sqlattack','xterm']
probe = ['ipsweep','mscan','nmap','portsweep','saint','satan']

class OutcomeCreater(Transformer): # this defines a transformer that creates the outcome

    def __init__(self):
        super().__init__()

    def _transform(self, dataset):
        label_to_binary = udf(lambda name: 0.0 if name == 'normal' else 1.0 if name in D
        label_to_category = udf(lambda name: attack_category[0] if name == 0.0 else atta
        output_df = dataset.withColumn('outcome', label_to_binary(col('class'))).drop("c
        output_df = output_df.withColumn('outcome', col('outcome').cast(DoubleType()))
        output_df = output_df.withColumn('outcome_category',label_to_category(col('outco
        output_df = output_df.withColumn('outcome_category', col('outcome_category').cas
        output_df = output_df.drop('difficulty')
        return output_df


class FeatureTypeCaster(Transformer): # this transformer will cast the columns as approp
    def __init__(self):
        super().__init__()

    def _transform(self, dataset):
        output_df = dataset
        for col_name in binary_cols + continuous_cols:
            output_df = output_df.withColumn(col_name,col(col_name).cast(DoubleType()))

        return output_df
class ColumnDropper(Transformer): # this transformer drops unnecessary columns
    def __init__(self, columns_to_drop = None):
        super().__init__()
        self.columns_to_drop=columns_to_drop
    def _transform(self, dataset):
        output_df = dataset
        for col_name in self.columns_to_drop:
            output_df = output_df.drop(col_name)
        return output_df

def get_preprocess_pipeline():
```

```
    # Stage where columns are casted as appropriate types
    stage_typecaster = FeatureTypeCaster()

    # Stage where nominal columns are transformed to index columns using StringIndexer
    nominal_id_cols = [x+"_index" for x in nominal_cols]
    nominal_onehot_cols = [x+"_encoded" for x in nominal_cols]
    stage_nominal_indexer = StringIndexer(inputCols = nominal_cols, outputCols = nominal

    # Stage where the index columns are further transformed using OneHotEncoder
    stage_nominal_onehot_encoder = OneHotEncoder(inputCols=nominal_id_cols, outputCols=n

    # Stage where all relevant features are assembled into a vector (and dropping a few)
    feature_cols = continuous_cols+binary_cols+nominal_onehot_cols
    corelated_cols_to_remove = ["dst_host_serror_rate","srv_serror_rate","dst_host_srv_s
                    "srv_rerror_rate","dst_host_rerror_rate","dst_host_srv_rerror_rate"
    for col_name in corelated_cols_to_remove:
        feature_cols.remove(col_name)
    stage_vector_assembler = VectorAssembler(inputCols=feature_cols, outputCol="vectoriz

    # Stage where we scale the columns
    stage_scaler = StandardScaler(inputCol= 'vectorized_features', outputCol= 'features'


    # Stage for creating the outcome column representing whether there is attack
    stage_outcome = OutcomeCreater()

    # Removing all unnecessary columbs, only keeping the 'features' and 'outcome' column
    stage_column_dropper = ColumnDropper(columns_to_drop = nominal_cols+nominal_id_cols+
        nominal_onehot_cols+ binary_cols + continuous_cols + ['vectorized_features'])
    # Connect the columns into a pipeline
    pipeline = Pipeline(stages=[stage_typecaster,stage_nominal_indexer,stage_nominal_one
        stage_vector_assembler,stage_scaler,stage_outcome,stage_column_dropper])
    return pipeline
```

In [38]:

```
import os
import sys

os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable

spark = SparkSession.builder \
    .master("local[*]") \
    .appName("SystemsToolChains") \
    .getOrCreate()

nslkdd_raw = spark.read.csv('/Users/kiranprasadjp/Downloads/NSL-KDD/KDDTrain+.txt',heade
nslkdd_test_raw = spark.read.csv('/Users/kiranprasadjp/Downloads/NSL-KDD/KDDTest+.txt',h

preprocess_pipeline = get_preprocess_pipeline()
preprocess_pipeline_model = preprocess_pipeline.fit(nslkdd_raw)

nslkdd_df = preprocess_pipeline_model.transform(nslkdd_raw)
nslkdd_df_test = preprocess_pipeline_model.transform(nslkdd_test_raw)
```

In [39]:

```
nslkdd_df.printSchema()
nslkdd_df.show()
```

```
root
 |-- features: vector (nullable = true)
 |-- outcome: double (nullable = true)
 |-- outcome_category: string (nullable = true)

+--------------------+-------+----------------+
|            features|outcome|outcome_category|
+--------------------+-------+----------------+
```

```
|(113,[1,13,14,17,...|    0.0|          normal|
|(113,[1,13,14,17,...|    0.0|          normal|
|(113,[13,14,15,17...|    1.0|             DOS|
|(113,[1,2,13,14,1...|    0.0|          normal|
|(113,[1,2,13,14,1...|    0.0|          normal|
|(113,[13,14,16,17...|    1.0|             DOS|
|(113,[13,14,15,17...|    1.0|             DOS|
|(113,[13,14,15,17...|    1.0|             DOS|
|(113,[13,14,15,17...|    1.0|             DOS|
|(113,[13,14,15,17...|    1.0|             DOS|
|(113,[13,14,16,17...|    1.0|             DOS|
|(113,[13,14,15,17...|    1.0|             DOS|
|(113,[1,2,13,14,1...|    0.0|          normal|
|(113,[1,13,14,17,...|    2.0|             R2L|
|(113,[13,14,15,18...|    1.0|             DOS|
|(113,[13,14,15,17...|    1.0|             DOS|
|(113,[1,2,13,14,1...|    0.0|          normal|
|(113,[1,13,14,17,...|    4.0|           probe|
|(113,[1,2,13,14,1...|    0.0|          normal|
|(113,[1,2,13,14,1...|    0.0|          normal|
+--------------------+-------+----------------+
only showing top 20 rows
```

In [40]:
```
to_array = udf(lambda v: v.toArray().tolist(), ArrayType(FloatType()))

nslkdd_df_train = nslkdd_df
nslkdd_df_validate,nslkdd_df_test = nslkdd_df_test.randomSplit([0.5,0.5])

nslkdd_df_train_pandas = nslkdd_df_train.withColumn('features', to_array('features')).to
nslkdd_df_validate_pandas = nslkdd_df_validate.withColumn('features', to_array('features
nslkdd_df_test_pandas = nslkdd_df_test.withColumn('features', to_array('features')).toPa
```

In [52]:
```
import tensorflow as tf
from tensorflow import keras

# Converting the pandas DataFrame to tensors
# Note we are using 3 data sets train, validate, test

x_train = tf.constant(np.array(nslkdd_df_train_pandas['features'].values.tolist()))
y_train = tf.constant(np.array(nslkdd_df_train_pandas['outcome'].values.tolist()))

x_validate = tf.constant(np.array(nslkdd_df_validate_pandas['features'].values.tolist())
y_validate = tf.constant(np.array(nslkdd_df_validate_pandas['outcome'].values.tolist()))


x_test = tf.constant(np.array(nslkdd_df_test_pandas['features'].values.tolist()))
y_test = tf.constant(np.array(nslkdd_df_test_pandas['outcome'].values.tolist()))
```

In [53]:
```
model = keras.Sequential( [keras.layers.Dense(10,activation='relu'),
                           keras.layers.Dense(10,activation='relu'),
                           keras.layers.Dense(10,activation='relu'),
                           keras.layers.Dense(10,activation='relu') ,
                           keras.layers.Dense(5)] )


y_pred = model(x_train)
model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
```

```
 dense_55 (Dense)             (125973, 10)              1140

 dense_56 (Dense)             (125973, 10)              110

 dense_57 (Dense)             (125973, 10)              110

 dense_58 (Dense)             (125973, 10)              110

 dense_59 (Dense)             (125973, 5)               55

=================================================================
Total params: 1,525
Trainable params: 1,525
Non-trainable params: 0
_________________________________________________________________
```

In [54]:
```
# Compile the model
model.compile(optimizer = 'sgd',
    loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True))
```

In [55]:
```
loss_func_from_logit_true = keras.losses.SparseCategoricalCrossentropy(from_logits=True)
loss_func_from_logit_false = keras.losses.SparseCategoricalCrossentropy(from_logits=Fals

loss_1 = loss_func_from_logit_true(y_train[:5],y_pred[:5])
loss_2 = loss_func_from_logit_false(y_train[:5],y_pred[:5])
loss_3 = loss_func_from_logit_false(y_train[:5],
                    tf.keras.activations.softmax(y_pred[:5]))  # This is correct as now

print("loss_1 = ", loss_1 )
print("loss_2 = ", loss_2 )
print("loss_3 = ", loss_3)
```

```
loss_1 =  tf.Tensor(1.5859284, shape=(), dtype=float32)
loss_2 =  tf.Tensor(6.242429, shape=(), dtype=float32)
loss_3 =  tf.Tensor(1.5859284, shape=(), dtype=float32)
```

In [57]:
```
model2 = keras.Sequential( [keras.layers.Dense(10,activation='relu'),
                            keras.layers.Dense(10,activation='relu'),
                            keras.layers.Dense(10,activation='relu'),
                            keras.layers.Dense(10,activation='relu') ,
                            keras.layers.Dense(5,activation='softmax')] )

model2.compile(optimizer = 'sgd',loss=keras.losses.SparseCategoricalCrossentropy(from_lo
model2.fit(x_train,y_train, epochs = 5,batch_size = 64, validation_data=(x_validate,y_va
```

```
Epoch 1/5
1969/1969 - 1s - loss: 0.4019 - val_loss: 1.0996 - 724ms/epoch - 368us/step
Epoch 2/5
1969/1969 - 1s - loss: 0.1294 - val_loss: 1.1572 - 524ms/epoch - 266us/step
Epoch 3/5
1969/1969 - 1s - loss: 0.1095 - val_loss: 1.2030 - 527ms/epoch - 267us/step
Epoch 4/5
1969/1969 - 1s - loss: 0.0984 - val_loss: 1.2118 - 583ms/epoch - 296us/step
Epoch 5/5
1969/1969 - 1s - loss: 0.0888 - val_loss: 1.2600 - 580ms/epoch - 295us/step
```

Out[57]:
```
<keras.callbacks.History at 0x28130ffa0>
```

In [58]:
```
import tensorflow as tf
print("Num CPUs Available: ", len(tf.config.experimental.list_physical_devices('CPU')))
print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU')))
```

```
Num CPUs Available:  1
Num GPUs Available:  0
```

In [59]:
```
!python --version
```

```
Python 3.11.4
```

In [60]:
```
print(tf.__version__)
```

```
2.11.0
```

In [ ]:
```
```

In [62]:
```
model = keras.Sequential( [keras.layers.Dense(10,activation='relu'),
                           keras.layers.Dense(10,activation='relu'),
                           keras.layers.Dense(10,activation='relu'),
                           keras.layers.Dense(10,activation='relu') ,
                           keras.layers.Dense(5)] )


model.compile(optimizer = keras.optimizers.SGD(learning_rate=0.02),
    loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=[keras.metrics.SparseCategoricalAccuracy()])

model.fit(x_train,y_train, epochs = 5,batch_size = 64, validation_data=(x_validate,y_val
```

```
Epoch 1/5
1969/1969 - 1s - loss: 0.1778 - sparse_categorical_accuracy: 0.9518 - val_loss: 1.3084 -
val_sparse_categorical_accuracy: 0.7314 - 763ms/epoch - 388us/step
Epoch 2/5
1969/1969 - 1s - loss: 0.0692 - sparse_categorical_accuracy: 0.9785 - val_loss: 1.4101 -
val_sparse_categorical_accuracy: 0.7532 - 567ms/epoch - 288us/step
Epoch 3/5
1969/1969 - 1s - loss: 0.0523 - sparse_categorical_accuracy: 0.9831 - val_loss: 1.6284 -
val_sparse_categorical_accuracy: 0.7602 - 549ms/epoch - 279us/step
Epoch 4/5
1969/1969 - 1s - loss: 0.0451 - sparse_categorical_accuracy: 0.9847 - val_loss: 1.6593 -
val_sparse_categorical_accuracy: 0.7747 - 556ms/epoch - 282us/step
Epoch 5/5
1969/1969 - 1s - loss: 0.0406 - sparse_categorical_accuracy: 0.9854 - val_loss: 1.8363 -
val_sparse_categorical_accuracy: 0.7694 - 549ms/epoch - 279us/step
```

Out[62]:
```
<keras.callbacks.History at 0x2f45e6320>
```

In [63]:
```
model.evaluate(x_test,y_test, verbose = 2)
```

```
353/353 - 0s - loss: 1.7125 - sparse_categorical_accuracy: 0.7699 - 104ms/epoch - 296us/
step
```

Out[63]:
```
[1.712549090385437, 0.7699272036552429]
```

In [67]:
```
model3 = keras.Sequential( [keras.layers.Dense(10,activation='relu'),
                           keras.layers.Dense(10,activation='relu'),
                           keras.layers.Dense(10,activation='relu'),
                           keras.layers.Dense(10,activation='relu') ,
                           keras.layers.Dense(5)] )
model3.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.001),  # Use Adam optimizer with a l
    loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=[keras.metrics.SparseCategoricalAccuracy()]
)
```

In [70]:
```
history = model3.fit(x_train,y_train, epochs = 5,batch_size = 64, validation_data=(x_val
```

```
Epoch 1/5
1969/1969 - 1s - loss: 0.0234 - sparse_categorical_accuracy: 0.9932 - val_loss: 2.2463 -
val_sparse_categorical_accuracy: 0.7647 - 649ms/epoch - 329us/step
Epoch 2/5
1969/1969 - 1s - loss: 0.0220 - sparse_categorical_accuracy: 0.9935 - val_loss: 2.4808 -
val_sparse_categorical_accuracy: 0.7491 - 612ms/epoch - 311us/step
Epoch 3/5
```

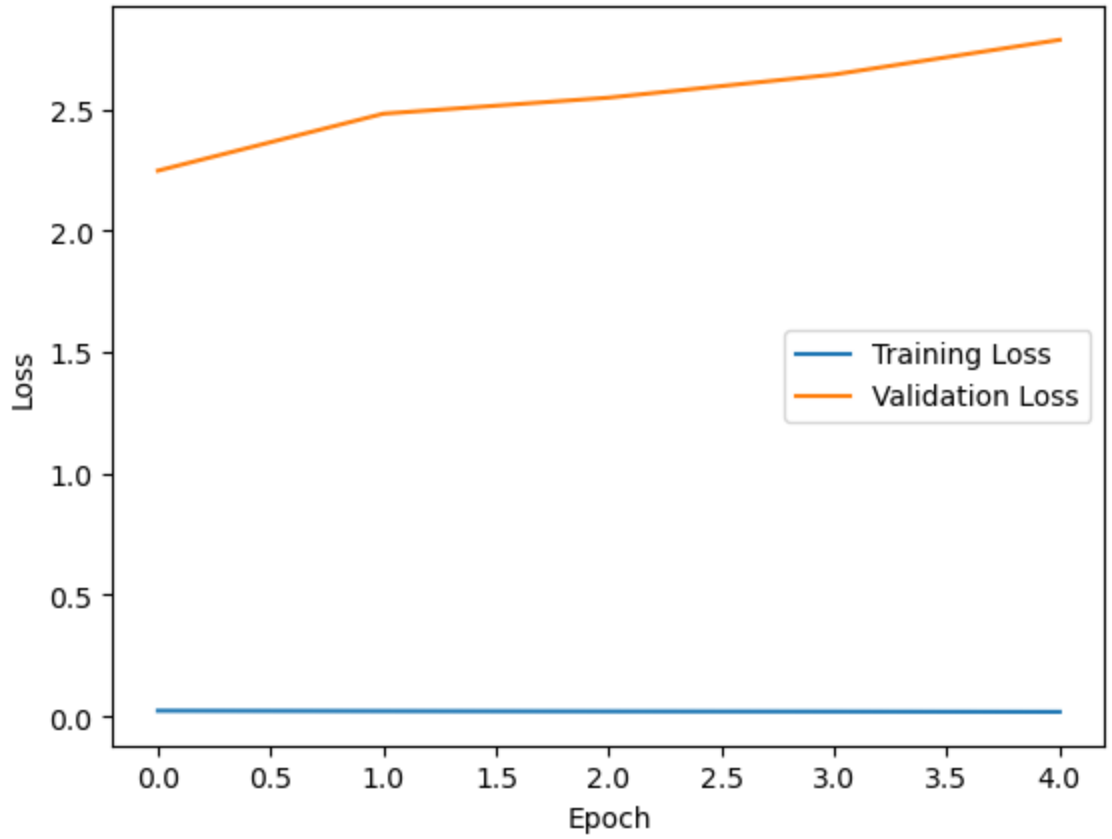
```
1969/1969 - 1s - loss: 0.0209 - sparse_categorical_accuracy: 0.9940 - val_loss: 2.5469 -
val_sparse_categorical_accuracy: 0.7509 - 614ms/epoch - 312us/step
Epoch 4/5
1969/1969 - 1s - loss: 0.0201 - sparse_categorical_accuracy: 0.9942 - val_loss: 2.6421 -
val_sparse_categorical_accuracy: 0.7572 - 616ms/epoch - 313us/step
Epoch 5/5
1969/1969 - 1s - loss: 0.0188 - sparse_categorical_accuracy: 0.9948 - val_loss: 2.7848 -
val_sparse_categorical_accuracy: 0.7388 - 613ms/epoch - 311us/step
```

In [71]:
```
model3.evaluate(x_test,y_test, verbose = 2)
```

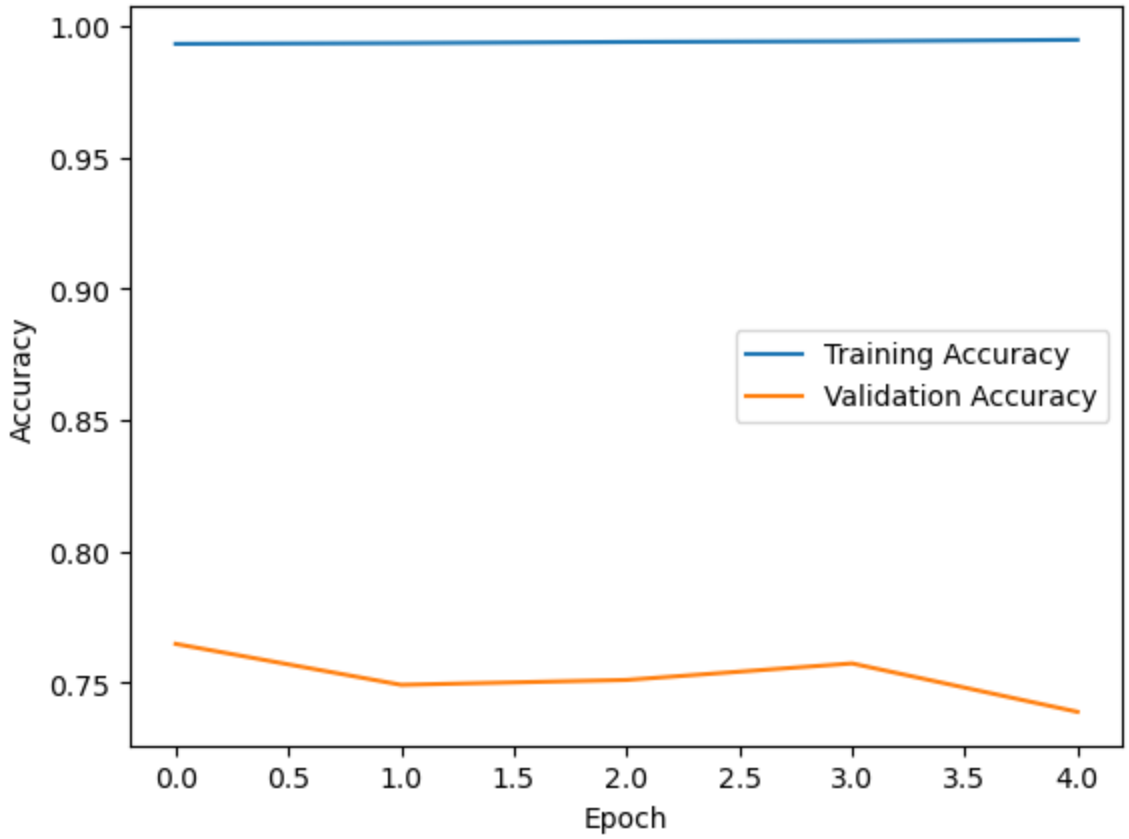
```
353/353 - 0s - loss: 2.6755 - sparse_categorical_accuracy: 0.7382 - 106ms/epoch - 299us/
step
```

Out[71]:
```
[2.6755471229553223, 0.7381501793861389]
```

In [73]:
```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [74]:
```
# Plot training history for accuracy
plt.plot(history.history['sparse_categorical_accuracy'], label='Training Accuracy')
plt.plot(history.history['val_sparse_categorical_accuracy'], label='Validation Accuracy'
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [79]:
```
!pip3 install tensorboard
```

```
Collecting tensorboard
  Downloading tensorboard-2.15.1-py3-none-any.whl.metadata (1.7 kB)
Collecting absl-py>=0.4 (from tensorboard)
  Downloading absl_py-2.0.0-py3-none-any.whl.metadata (2.3 kB)
Collecting grpcio>=1.48.2 (from tensorboard)
  Downloading grpcio-1.59.3-cp311-cp311-macosx_10_10_universal2.whl.metadata (4.0 kB)
Collecting google-auth<3,>=1.6.3 (from tensorboard)
  Downloading google_auth-2.23.4-py2.py3-none-any.whl.metadata (4.7 kB)
Collecting google-auth-oauthlib<2,>=0.5 (from tensorboard)
  Downloading google_auth_oauthlib-1.1.0-py2.py3-none-any.whl.metadata (2.7 kB)
Collecting markdown>=2.6.8 (from tensorboard)
  Downloading Markdown-3.5.1-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: numpy>=1.12.0 in /Applications/ANACONDA/anaconda3/envs/aiml_envv/lib/python3.11/site-packages (from tensorboard) (1.26.0)
Collecting protobuf<4.24,>=3.19.6 (from tensorboard)
  Downloading protobuf-4.23.4-cp37-abi3-macosx_10_9_universal2.whl.metadata (540 bytes)
Requirement already satisfied: requests<3,>=2.21.0 in /Applications/ANACONDA/anaconda3/envs/aiml_envv/lib/python3.11/site-packages (from tensorboard) (2.31.0)
Requirement already satisfied: setuptools>=41.0.0 in /Applications/ANACONDA/anaconda3/envs/aiml_envv/lib/python3.11/site-packages (from tensorboard) (68.0.0)
Requirement already satisfied: six>1.9 in /Applications/ANACONDA/anaconda3/envs/aiml_envv/lib/python3.11/site-packages (from tensorboard) (1.16.0)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard)
  Downloading tensorboard_data_server-0.7.2-py3-none-any.whl.metadata (1.1 kB)
Collecting werkzeug>=1.0.1 (from tensorboard)
  Downloading werkzeug-3.0.1-py3-none-any.whl.metadata (4.1 kB)
Collecting cachetools<6.0,>=2.0.0 (from google-auth<3,>=1.6.3->tensorboard)
  Downloading cachetools-5.3.2-py3-none-any.whl.metadata (5.2 kB)
Collecting pyasn1-modules>=0.2.1 (from google-auth<3,>=1.6.3->tensorboard)
  Downloading pyasn1_modules-0.3.0-py2.py3-none-any.whl (181 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 181.3/181.3 kB 1.9 MB/s eta 0:00:00a 0:00:01
Collecting rsa<5,>=3.1.4 (from google-auth<3,>=1.6.3->tensorboard)
  Downloading rsa-4.9-py3-none-any.whl (34 kB)
Collecting requests-oauthlib>=0.7.0 (from google-auth-oauthlib<2,>=0.5->tensorboard)
```

```
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /Applications/ANACONDA/anacon
da3/envs/aiml_envv/lib/python3.11/site-packages (from requests<3,>=2.21.0->tensorboard)
 (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /Applications/ANACONDA/anaconda3/envs/aim
l_envv/lib/python3.11/site-packages (from requests<3,>=2.21.0->tensorboard) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Applications/ANACONDA/anaconda3/en
vs/aiml_envv/lib/python3.11/site-packages (from requests<3,>=2.21.0->tensorboard) (1.26.
16)
Requirement already satisfied: certifi>=2017.4.17 in /Applications/ANACONDA/anaconda3/en
vs/aiml_envv/lib/python3.11/site-packages (from requests<3,>=2.21.0->tensorboard) (2023.
7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in /Applications/ANACONDA/anaconda3/env
s/aiml_envv/lib/python3.11/site-packages (from werkzeug>=1.0.1->tensorboard) (2.1.1)
Collecting pyasn1<0.6.0,>=0.4.6 (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tens
orboard)
  Downloading pyasn1-0.5.1-py2.py3-none-any.whl.metadata (8.6 kB)
Collecting oauthlib>=3.0.0 (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5-
>tensorboard)
  Downloading oauthlib-3.2.2-py3-none-any.whl (151 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 151.7/151.7 kB 2.0 MB/s eta 0:00:00a 0:00:0
1
Downloading tensorboard-2.15.1-py3-none-any.whl (5.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 3.2 MB/s eta 0:00:0000:0100:01
Downloading absl_py-2.0.0-py3-none-any.whl (130 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 130.2/130.2 kB 4.9 MB/s eta 0:00:00
Downloading google_auth-2.23.4-py2.py3-none-any.whl (183 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 183.3/183.3 kB 3.8 MB/s eta 0:00:00a 0:00:01
Downloading google_auth_oauthlib-1.1.0-py2.py3-none-any.whl (19 kB)
Downloading grpcio-1.59.3-cp311-cp311-macosx_10_10_universal2.whl (9.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.6/9.6 MB 5.3 MB/s eta 0:00:0000:0100:01
Downloading Markdown-3.5.1-py3-none-any.whl (102 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 102.2/102.2 kB 5.7 MB/s eta 0:00:00
Downloading protobuf-4.23.4-cp37-abi3-macosx_10_9_universal2.whl (400 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 400.3/400.3 kB 5.6 MB/s eta 0:00:00a 0:00:01
Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Downloading werkzeug-3.0.1-py3-none-any.whl (226 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 226.7/226.7 kB 5.1 MB/s eta 0:00:0000:01
Downloading cachetools-5.3.2-py3-none-any.whl (9.3 kB)
Downloading pyasn1-0.5.1-py2.py3-none-any.whl (84 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.9/84.9 kB 11.2 MB/s eta 0:00:00
Installing collected packages: werkzeug, tensorboard-data-server, pyasn1, protobuf, oaut
hlib, markdown, grpcio, cachetools, absl-py, rsa, requests-oauthlib, pyasn1-modules, goo
gle-auth, google-auth-oauthlib, tensorboard
Successfully installed absl-py-2.0.0 cachetools-5.3.2 google-auth-2.23.4 google-auth-oau
thlib-1.1.0 grpcio-1.59.3 markdown-3.5.1 oauthlib-3.2.2 protobuf-4.23.4 pyasn1-0.5.1 pya
sn1-modules-0.3.0 requests-oauthlib-1.3.1 rsa-4.9 tensorboard-2.15.1 tensorboard-data-se
rver-0.7.2 werkzeug-3.0.1
```

In [80]:
```
!pip3 show tensorboard
```

```
Name: tensorboard
Version: 2.15.1
Summary: TensorBoard lets you watch Tensors Flow
Home-page: https://github.com/tensorflow/tensorboard
Author: Google Inc.
Author-email: packages@tensorflow.org
License: Apache 2.0
Location: /Applications/ANACONDA/anaconda3/envs/aiml_envv/lib/python3.11/site-packages
Requires: absl-py, google-auth, google-auth-oauthlib, grpcio, markdown, numpy, protobuf,
requests, setuptools, six, tensorboard-data-server, werkzeug
Required-by:
```

In [ ]:
```
python3 /Applications/ANACONDA/anaconda3/envs/aiml_envv/lib/python3.11/site-packages/ten
```

In [98]:
```
import datetime

model = keras.Sequential( [keras.layers.Dense(10,activation='relu'),
    keras.layers.Dense(10,activation='relu'),
    keras.layers.Dense(10,activation='relu'),
    keras.layers.Dense(10,activation='relu') ,
    keras.layers.Dense(5)] )
model.compile(loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
        metrics=[keras.metrics.SparseCategoricalAccuracy(name='Accuracy')])
log_dir = "./logs14763/myfirstlog/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
model.fit(x=x_train, y=y_train,
          epochs=20, verbose = 2,
          validation_data=(x_validate, y_validate),
          callbacks=[tensorboard_callback])
```

```
Epoch 1/20
3937/3937 - 1s - loss: 0.1360 - Accuracy: 0.9586 - val_loss: 1.9907 - val_Accuracy: 0.72
97 - 1s/epoch - 337us/step
Epoch 2/20
3937/3937 - 1s - loss: 0.0502 - Accuracy: 0.9866 - val_loss: 2.4656 - val_Accuracy: 0.73
72 - 1s/epoch - 276us/step
Epoch 3/20
3937/3937 - 1s - loss: 0.0431 - Accuracy: 0.9887 - val_loss: 2.5710 - val_Accuracy: 0.73
76 - 1s/epoch - 281us/step
Epoch 4/20
3937/3937 - 1s - loss: 0.0397 - Accuracy: 0.9896 - val_loss: 2.7243 - val_Accuracy: 0.73
38 - 1s/epoch - 278us/step
Epoch 5/20
3937/3937 - 1s - loss: 0.0370 - Accuracy: 0.9906 - val_loss: 3.0689 - val_Accuracy: 0.73
87 - 1s/epoch - 279us/step
Epoch 6/20
3937/3937 - 1s - loss: 0.0364 - Accuracy: 0.9912 - val_loss: 3.2086 - val_Accuracy: 0.73
96 - 1s/epoch - 285us/step
Epoch 7/20
3937/3937 - 1s - loss: 0.0357 - Accuracy: 0.9914 - val_loss: 2.8019 - val_Accuracy: 0.73
64 - 1s/epoch - 283us/step
Epoch 8/20
3937/3937 - 1s - loss: 0.0357 - Accuracy: 0.9916 - val_loss: 2.6877 - val_Accuracy: 0.74
74 - 1s/epoch - 278us/step
Epoch 9/20
3937/3937 - 1s - loss: 0.0351 - Accuracy: 0.9919 - val_loss: 3.5533 - val_Accuracy: 0.73
69 - 1s/epoch - 279us/step
Epoch 10/20
3937/3937 - 1s - loss: 0.0357 - Accuracy: 0.9921 - val_loss: 3.3714 - val_Accuracy: 0.75
10 - 1s/epoch - 292us/step
Epoch 11/20
3937/3937 - 1s - loss: 0.0381 - Accuracy: 0.9920 - val_loss: 3.2866 - val_Accuracy: 0.74
49 - 1s/epoch - 282us/step
Epoch 12/20
3937/3937 - 1s - loss: 0.0400 - Accuracy: 0.9922 - val_loss: 2.8069 - val_Accuracy: 0.74
62 - 1s/epoch - 278us/step
Epoch 13/20
3937/3937 - 1s - loss: 0.0452 - Accuracy: 0.9922 - val_loss: 3.4856 - val_Accuracy: 0.74
56 - 1s/epoch - 276us/step
Epoch 14/20
3937/3937 - 1s - loss: 0.0521 - Accuracy: 0.9917 - val_loss: 3.2027 - val_Accuracy: 0.74
82 - 1s/epoch - 276us/step
Epoch 15/20
3937/3937 - 1s - loss: 0.0518 - Accuracy: 0.9917 - val_loss: 3.8235 - val_Accuracy: 0.74
54 - 1s/epoch - 278us/step
Epoch 16/20
3937/3937 - 1s - loss: 0.0550 - Accuracy: 0.9920 - val_loss: 3.9655 - val_Accuracy: 0.74
64 - 1s/epoch - 283us/step
Epoch 17/20
3937/3937 - 1s - loss: 0.0571 - Accuracy: 0.9922 - val_loss: 4.8482 - val_Accuracy: 0.74
```

```
37 - 1s/epoch - 278us/step
Epoch 18/20
3937/3937 - 1s - loss: 0.0558 - Accuracy: 0.9923 - val_loss: 3.9925 - val_Accuracy: 0.74
30 - 1s/epoch - 305us/step
Epoch 19/20
3937/3937 - 1s - loss: 0.0582 - Accuracy: 0.9922 - val_loss: 3.8774 - val_Accuracy: 0.74
29 - 1s/epoch - 310us/step
Epoch 20/20
3937/3937 - 1s - loss: 0.0645 - Accuracy: 0.9922 - val_loss: 4.2170 - val_Accuracy: 0.73
90 - 1s/epoch - 286us/step
```

Out[98]: <keras.callbacks.History at 0x13fd59e10>

In [ ]: