AutoML Models for Modulation Classification

March 24, 2021

1 Modulation Classification

Developing AutoML Model for AWGN Channel for Modulation Classification. SNR Ratio's(in dB) of Channel are $[5, 10, 15, 20, 25, 30]$.

Data Generation: Data is generated using MatLab's Communication Toolbox for Modulation Schemes QPSK, 16-QAM, 64-QAM for SNR Ratio's (in dB) $[5, 10, 15, 20, 25, 30]$ when signal is transmitted through Channels AWGN and Rayleigh Channels.

Modulation Classification: We will use AutoML to creae a classifer that predicts Modulation Scheme depending on In-Phase and Quadrature-Phase Components at the Receiver's End.

1.1 Imports

1.1.1 Importing Libraries

We will be using AutoKeras for generating AutoML Models. Source of Documentation: https://autokeras.com/

[1]:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as pimg
import seaborn as sns
import scipy.io
import os

# AutoML Libraries
!pip3 install autokeras
import autokeras as ak

# Tensorflow Libraries
import tensorflow as tf
```

```
Requirement already satisfied: autokeras in /usr/local/lib/python3.7/dist-
packages (1.0.12)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages
(from autokeras) (1.1.5)
Requirement already satisfied: keras-tuner>=1.0.2 in
```

```
/usr/local/lib/python3.7/dist-packages (from autokeras) (1.0.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from autokeras) (20.9)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-
packages (from autokeras) (0.22.2.post1)
Requirement already satisfied: tensorflow>=2.3.0 in
/usr/local/lib/python3.7/dist-packages (from autokeras) (2.4.1)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-
packages (from pandas->autokeras) (1.19.5)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.7/dist-packages (from pandas->autokeras) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-
packages (from pandas->autokeras) (2018.9)
Requirement already satisfied: colorama in /usr/local/lib/python3.7/dist-
packages (from keras-tuner>=1.0.2->autokeras) (0.4.4)
Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-
packages (from keras-tuner>=1.0.2->autokeras) (0.8.9)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
(from keras-tuner>=1.0.2->autokeras) (4.41.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from keras-tuner>=1.0.2->autokeras) (2.23.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from keras-tuner>=1.0.2->autokeras) (1.4.1)
Requirement already satisfied: terminaltables in /usr/local/lib/python3.7/dist-
packages (from keras-tuner>=1.0.2->autokeras) (3.1.0)
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages
(from keras-tuner>=1.0.2->autokeras) (0.16.0)
Requirement already satisfied: pyparsing>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging->autokeras) (2.4.7)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn->autokeras) (1.0.1)
Requirement already satisfied: astunparse~=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.3.0->autokeras)
(1.6.3)
Requirement already satisfied: six~=1.15.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.3.0->autokeras) (1.15.0)
Requirement already satisfied: h5py~=2.10.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.3.0->autokeras) (2.10.0)
Requirement already satisfied: tensorboard~=2.4 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.3.0->autokeras)
(2.4.1)
Requirement already satisfied: grpcio~=1.32.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.3.0->autokeras) (1.32.0)
Requirement already satisfied: flatbuffers~=1.12.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.3.0->autokeras)
(1.12)
Requirement already satisfied: keras-preprocessing~=1.1.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.3.0->autokeras)
```

```
(1.1.2)
Requirement already satisfied: termcolor~=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.3.0->autokeras)
(1.1.0)
Requirement already satisfied: typing-extensions~=3.7.4 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.3.0->autokeras)
(3.7.4.3)
Requirement already satisfied: google-pasta~=0.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.3.0->autokeras)
(0.2.0)
Requirement already satisfied: wrapt~=1.12.1 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.3.0->autokeras) (1.12.1)
Requirement already satisfied: opt-einsum~=3.3.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.3.0->autokeras)
(3.3.0)
Requirement already satisfied: wheel~=0.35 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.3.0->autokeras) (0.36.2)
Requirement already satisfied: absl-py~=0.10 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.3.0->autokeras) (0.10.0)
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.3.0->autokeras) (3.12.4)
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.3.0->autokeras) (0.3.3)
Requirement already satisfied: tensorflow-estimator<2.5.0,>=2.4.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.3.0->autokeras)
(2.4.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests->keras-tuner>=1.0.2->autokeras) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests->keras-
tuner>=1.0.2->autokeras) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests->keras-
tuner>=1.0.2->autokeras) (2020.12.5)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests->keras-
tuner>=1.0.2->autokeras) (3.0.4)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (54.1.2)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (1.0.1)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (1.27.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (0.4.3)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (1.8.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-
packages (from tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (3.3.4)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<2,>=1.6.3->tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3.6" in
/usr/local/lib/python3.7/dist-packages (from google-
auth<2,>=1.6.3->tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<2,>=1.6.3->tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (4.2.1)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (1.3.0)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in
/usr/local/lib/python3.7/dist-packages (from
markdown>=2.6.8->tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (3.7.2)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-
auth<2,>=1.6.3->tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-
packages (from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (3.1.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata; python_version <
"3.8"->markdown>=2.6.8->tensorboard~=2.4->tensorflow>=2.3.0->autokeras) (3.4.1)
```

1.1.2 Importing Data

Files are uploaded in Google Drive. Notebook is connected to Google Drive

[2]:

```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive/My\ Drive/Modulation-Classification/AutoML
```

```
Drive already mounted at /gdrive; to attempt to forcibly remount, call
drive.mount("/gdrive", force_remount=True).
/gdrive/My Drive/Modulation-Classification/AutoML
```

1.2 Processsing Data

Importing Data for all SNR Ratio’s

[3]:

```python
def ImportData(Path):
    ModulationSchemes = os.listdir(Path)

    Dataset = {}
    for scheme in ModulationSchemes:
        DataPath = Path + "/" + scheme
        files = os.listdir(DataPath)
        DataofScheme = {}
        for f in files:
            Data = scipy.io.loadmat(DataPath + "/" + f)
            snr = Data['snr'][0][0]
            rx = Data['rx']
            InPhase, QuadPhase = rx.real, rx.imag
            DataofScheme[snr] = np.append(InPhase, QuadPhase, axis=1)
        Dataset[scheme] = DataofScheme

    return Dataset
```

1.3 Train and Validation Datasets

Training: - Received Signal with SNR Ratio 30dB is used for Training for both Channels and all Modulation Schemes.

Validation: - AutoML is validated on Received Signals with SNR Ratio's (in dB) $[5, 10, 15, 20, 25]$

[4]:

```python
def GenerateDatasets(Channel,L=None):
    if Channel == "AWGN":
        Path = "../Data/" + Channel
    elif Channel == "Rayleigh":
        Path = "../Data/" + Channel + "/" + str(L)
    Data = ImportData(Path)

    Dataset = {}
    Dataset['Classes'] = list(Data.keys())
    OneHotClasses = np.eye(len(Dataset['Classes']))

    Classes = {}
    for i in range(len(Dataset['Classes'])):
        Classes[Dataset['Classes'][i]] = OneHotClasses[i]

    Valid_SNRs = [5,10,15,20,25]

    if Channel == "AWGN":
        X_Train, y_Train = np.empty((0,2)), np.empty((0,3))
        X_Valid, y_Valid = {}, {}

        for snr in Valid_SNRs:
            X_Valid[snr] = np.empty((0,2))
            y_Valid[snr] = np.empty((0,3))


        for c in Classes.keys():
            ModData = Data[c]
            SNRs = ModData.keys()
            for snr in SNRs:
                if snr == 30:
                    X = ModData[snr]
                    y = np.repeat(np.expand_dims(Classes[c],axis=0), X.
↪shape[0], axis=0)
                    X_Train = np.append(X_Train,X,axis=0)
                    y_Train = np.append(y_Train,y,axis=0)
                else:
                    X = ModData[snr]
                    y = np.repeat(np.expand_dims(Classes[c],axis=0), X.
↪shape[0], axis=0)
                    X_Valid[snr] = np.append(X_Valid[snr], X, axis=0)
                    y_Valid[snr] = np.append(y_Valid[snr], y, axis=0)

    elif Channel == "Rayleigh":
        X_Train, y_Train = np.empty((0,100,2)), np.empty((0,3))
        X_Valid, y_Valid = {}, {}
        for snr in Valid_SNRs:
            X_Valid[snr] = np.empty((0,100,2))
            y_Valid[snr] = np.empty((0,3))

        for c in Classes.keys():
            ModData = Data[c]
            SNRs = ModData.keys()
            for snr in SNRs:
                if snr == 30:
                    X = ModData[snr]
                    X = X.reshape(-1,100,2)
                    y = np.repeat(np.expand_dims(Classes[c],axis=0), X.
↪shape[0], axis=0)
                    X_Train = np.append(X_Train,X,axis=0)
                    y_Train = np.append(y_Train,y,axis=0)
                else:
                    X = ModData[snr]
                    X = X.reshape(-1,100,2)
                    y = np.repeat(np.expand_dims(Classes[c],axis=0), X.
↪shape[0], axis=0)
                    X_Valid[snr] = np.append(X_Valid[snr], X, axis=0)
                    y_Valid[snr] = np.append(y_Valid[snr], y, axis=0)
    return X_Train, y_Train, X_Valid, y_Valid
```

1.4 Evaluating Data

[5]:

```
def EvaluateData(Model, X_Train, y_Train, X_Valid, y_Valid, Name):
    Valid_SNR = np.array([5,10,15,20,25,30])
    Accuracy = []

    print ("Evaluating Model")
    for snr in Valid_SNR:
        if snr == 30:
            Loss, Acc = Model.evaluate(X_Train, y_Train)
        else:
            Loss, Acc = Model.evaluate(X_Valid[snr], y_Valid[snr])
        print ("SNR:", snr, "Accuracy:", Acc)
        Accuracy.append(Acc)

    Accuracy = np.array(Accuracy)

    plt.figure(figsize=(10,10))
    plt.plot(Valid_SNR,Accuracy, color='blue')
    plt.scatter(Valid_SNR,Accuracy, color='red')
    plt.title("Accuracy vs SNR")
    plt.xlabel("SNR")
    plt.ylabel("Accuracy")
    plt.grid()
    plt.savefig("Images/" + Name)
    plt.show()
```

1.5 AutoML Model for AWGN Channel

For AWGN Channel, Input Dimensions is (2,)

AWGN Data

[6]:

```
X_Train, y_Train, X_Valid, y_Valid = GenerateDatasets('AWGN')
```

Creating a Classifier

[7]:

```
AWGNClassifier = ak.StructuredDataClassifier(
    overwrite=True,
    max_trials=10,
    loss='categorical_crossentropy',
    num_classes=3)
```

```python
AWGNClassifier.fit(X_Train, y_Train, epochs=15, batch_size=64,␣
 ↪validation_split=0.1)
```

```
Trial 10 Complete [00h 00m 19s]
val_accuracy: 1.0

Best val_accuracy So Far: 1.0
Total elapsed time: 00h 03m 05s
INFO:tensorflow:Oracle triggered exit
Epoch 1/15
469/469 [==============================] - 2s 2ms/step - loss: 0.9269 -
accuracy: 0.8846
Epoch 2/15
469/469 [==============================] - 1s 2ms/step - loss: 1.2889 -
accuracy: 0.4880
Epoch 3/15
469/469 [==============================] - 1s 2ms/step - loss: 0.8460 -
accuracy: 0.6703
Epoch 4/15
469/469 [==============================] - 1s 2ms/step - loss: 0.7505 -
accuracy: 0.7085
Epoch 5/15
469/469 [==============================] - 1s 2ms/step - loss: 0.6694 -
accuracy: 0.7330
Epoch 6/15
469/469 [==============================] - 1s 2ms/step - loss: 0.6282 -
accuracy: 0.7421
Epoch 7/15
469/469 [==============================] - 1s 2ms/step - loss: 0.5567 -
accuracy: 0.7586
Epoch 8/15
469/469 [==============================] - 1s 2ms/step - loss: 0.5165 -
accuracy: 0.7678
Epoch 9/15
469/469 [==============================] - 1s 2ms/step - loss: 0.5981 -
accuracy: 0.7578
Epoch 10/15
469/469 [==============================] - 1s 2ms/step - loss: 0.4665 -
accuracy: 0.7864
Epoch 11/15
469/469 [==============================] - 1s 2ms/step - loss: 0.4329 -
accuracy: 0.7991
Epoch 12/15
469/469 [==============================] - 1s 2ms/step - loss: 0.3973 -
accuracy: 0.8129
Epoch 13/15
469/469 [==============================] - 1s 2ms/step - loss: 0.3581 -
```

```
accuracy: 0.8296
Epoch 14/15
469/469 [==============================] - 1s 2ms/step - loss: 0.3219 -
accuracy: 0.8450
Epoch 15/15
469/469 [==============================] - 1s 2ms/step - loss: 0.2883 -
accuracy: 0.8602
INFO:tensorflow:Assets written to:
./structured_data_classifier/best_model/assets
```

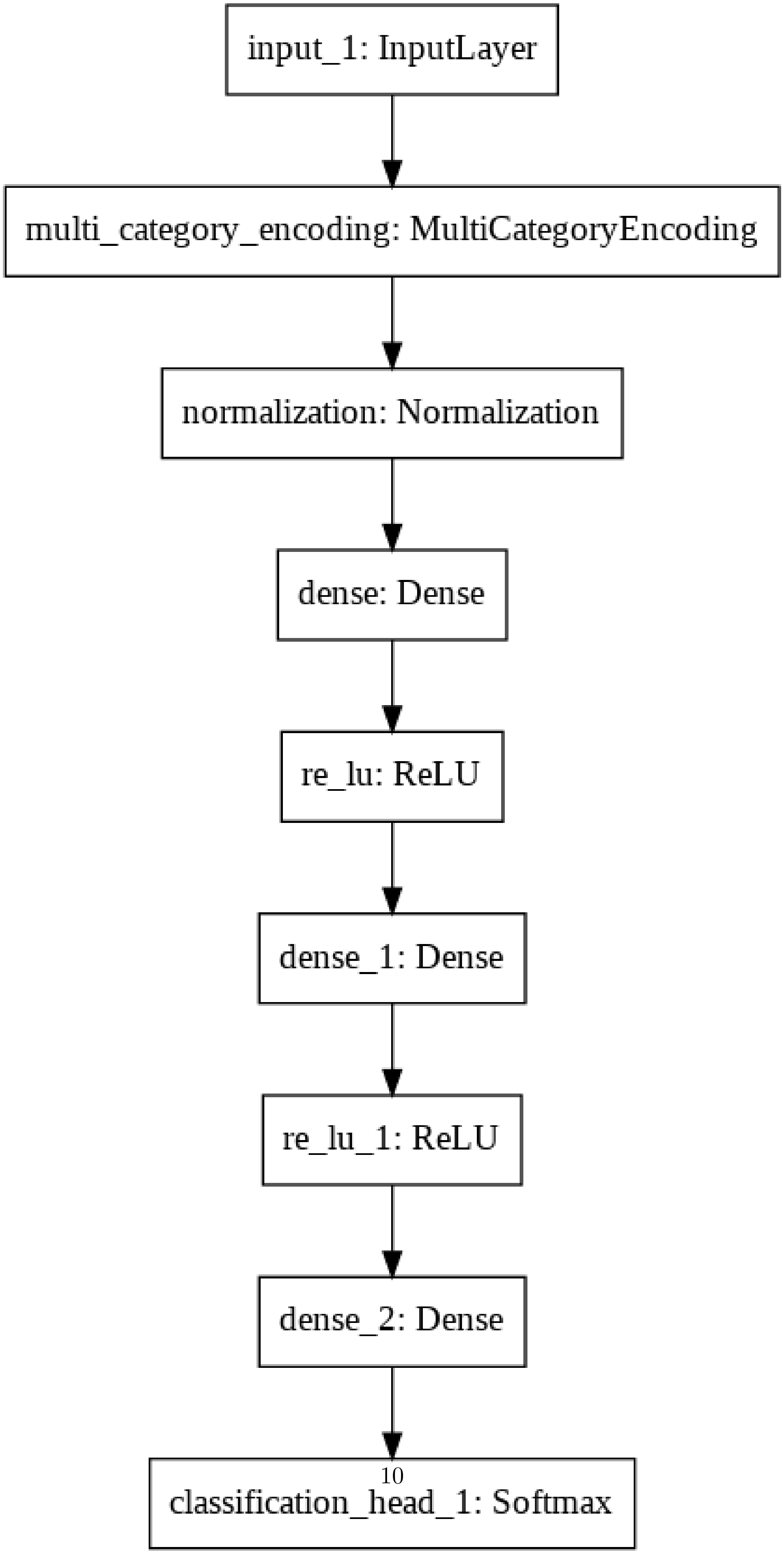
Model Summary

[8]:

```
AWGN_Model = AWGNClassifier.export_model()
AWGN_Model.summary()
tf.keras.utils.plot_model(AWGN_Model, to_file='Images/AWGN_Model.png',␣
 ↪show_shapes=False,show_layer_names=True)
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
multi_category_encoding (Mul (None, 2)                 0
_________________________________________________________________
normalization (Normalization (None, 2)                 5
_________________________________________________________________
dense (Dense)                (None, 32)                96
_________________________________________________________________
re_lu (ReLU)                 (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                1056
_________________________________________________________________
re_lu_1 (ReLU)               (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 99
_________________________________________________________________
classification_head_1 (Softm (None, 3)                 0
=================================================================
Total params: 1,256
Trainable params: 1,251
Non-trainable params: 5
_________________________________________________________________
```

[8]:


input_1: InputLayer
multi_category_encoding: MultiCategoryEncoding
normalization: Normalization
dense: Dense
re_lu: ReLU
dense_1: Dense
re_lu_1: ReLU
dense_2: Dense
classification_head_1: Softmax

1.5.1 Training and Evaluating Model

Training the Model

```
[9]: AWGN_Model.fit(X_Train, y_Train, epochs=50, batch_size=32, validation_split=0.
      ↪1,shuffle=True)
```

```
Epoch 1/50
844/844 [==============================] - 2s 3ms/step - loss: 0.2908 -
accuracy: 0.8854 - val_loss: 0.3373 - val_accuracy: 1.0000
Epoch 2/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2221 -
accuracy: 0.9057 - val_loss: 0.3141 - val_accuracy: 1.0000
Epoch 3/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2115 -
accuracy: 0.9062 - val_loss: 0.3142 - val_accuracy: 1.0000
Epoch 4/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2075 -
accuracy: 0.9066 - val_loss: 0.2998 - val_accuracy: 1.0000
Epoch 5/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2063 -
accuracy: 0.9065 - val_loss: 0.3199 - val_accuracy: 1.0000
Epoch 6/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2055 -
accuracy: 0.9065 - val_loss: 0.2924 - val_accuracy: 1.0000
Epoch 7/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2054 -
accuracy: 0.9064 - val_loss: 0.2741 - val_accuracy: 1.0000
Epoch 8/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2052 -
accuracy: 0.9065 - val_loss: 0.3102 - val_accuracy: 1.0000
Epoch 9/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2049 -
accuracy: 0.9065 - val_loss: 0.3277 - val_accuracy: 1.0000
Epoch 10/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2047 -
accuracy: 0.9064 - val_loss: 0.3357 - val_accuracy: 1.0000
Epoch 11/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2048 -
accuracy: 0.9065 - val_loss: 0.3179 - val_accuracy: 1.0000
Epoch 12/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2044 -
accuracy: 0.9066 - val_loss: 0.3423 - val_accuracy: 1.0000
Epoch 13/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2044 -
```

```
accuracy: 0.9065 - val_loss: 0.3226 - val_accuracy: 1.0000
Epoch 14/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2046 -
accuracy: 0.9066 - val_loss: 0.3269 - val_accuracy: 1.0000
Epoch 15/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2046 -
accuracy: 0.9065 - val_loss: 0.2747 - val_accuracy: 1.0000
Epoch 16/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2043 -
accuracy: 0.9066 - val_loss: 0.3025 - val_accuracy: 1.0000
Epoch 17/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2044 -
accuracy: 0.9066 - val_loss: 0.2811 - val_accuracy: 1.0000
Epoch 18/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2044 -
accuracy: 0.9064 - val_loss: 0.3501 - val_accuracy: 1.0000
Epoch 19/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2045 -
accuracy: 0.9066 - val_loss: 0.3346 - val_accuracy: 1.0000
Epoch 20/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2043 -
accuracy: 0.9065 - val_loss: 0.3080 - val_accuracy: 1.0000
Epoch 21/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2040 -
accuracy: 0.9066 - val_loss: 0.3266 - val_accuracy: 1.0000
Epoch 22/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2043 -
accuracy: 0.9061 - val_loss: 0.2784 - val_accuracy: 1.0000
Epoch 23/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2041 -
accuracy: 0.9066 - val_loss: 0.2737 - val_accuracy: 1.0000
Epoch 24/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2045 -
accuracy: 0.9064 - val_loss: 0.3054 - val_accuracy: 1.0000
Epoch 25/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2042 -
accuracy: 0.9066 - val_loss: 0.3233 - val_accuracy: 1.0000
Epoch 26/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2041 -
accuracy: 0.9065 - val_loss: 0.3178 - val_accuracy: 1.0000
Epoch 27/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2041 -
accuracy: 0.9066 - val_loss: 0.2944 - val_accuracy: 1.0000
Epoch 28/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2042 -
accuracy: 0.9065 - val_loss: 0.3252 - val_accuracy: 1.0000
Epoch 29/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2044 -
```

```
accuracy: 0.9066 - val_loss: 0.3275 - val_accuracy: 1.0000
Epoch 30/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2042 -
accuracy: 0.9064 - val_loss: 0.3146 - val_accuracy: 1.0000
Epoch 31/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2040 -
accuracy: 0.9065 - val_loss: 0.2944 - val_accuracy: 0.9997
Epoch 32/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2043 -
accuracy: 0.9064 - val_loss: 0.3273 - val_accuracy: 0.9937
Epoch 33/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2044 -
accuracy: 0.9064 - val_loss: 0.2853 - val_accuracy: 1.0000
Epoch 34/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2041 -
accuracy: 0.9064 - val_loss: 0.2936 - val_accuracy: 1.0000
Epoch 35/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2040 -
accuracy: 0.9067 - val_loss: 0.3164 - val_accuracy: 1.0000
Epoch 36/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2041 -
accuracy: 0.9066 - val_loss: 0.3026 - val_accuracy: 1.0000
Epoch 37/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2040 -
accuracy: 0.9064 - val_loss: 0.2883 - val_accuracy: 1.0000
Epoch 38/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2042 -
accuracy: 0.9064 - val_loss: 0.2828 - val_accuracy: 1.0000
Epoch 39/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2042 -
accuracy: 0.9065 - val_loss: 0.3158 - val_accuracy: 1.0000
Epoch 40/50
844/844 [==============================] - 2s 3ms/step - loss: 0.2041 -
accuracy: 0.9065 - val_loss: 0.3048 - val_accuracy: 0.9997
Epoch 41/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2041 -
accuracy: 0.9065 - val_loss: 0.2796 - val_accuracy: 1.0000
Epoch 42/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2041 -
accuracy: 0.9064 - val_loss: 0.3013 - val_accuracy: 1.0000
Epoch 43/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2040 -
accuracy: 0.9064 - val_loss: 0.3590 - val_accuracy: 0.9860
Epoch 44/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2041 -
accuracy: 0.9064 - val_loss: 0.3004 - val_accuracy: 1.0000
Epoch 45/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2038 -
```

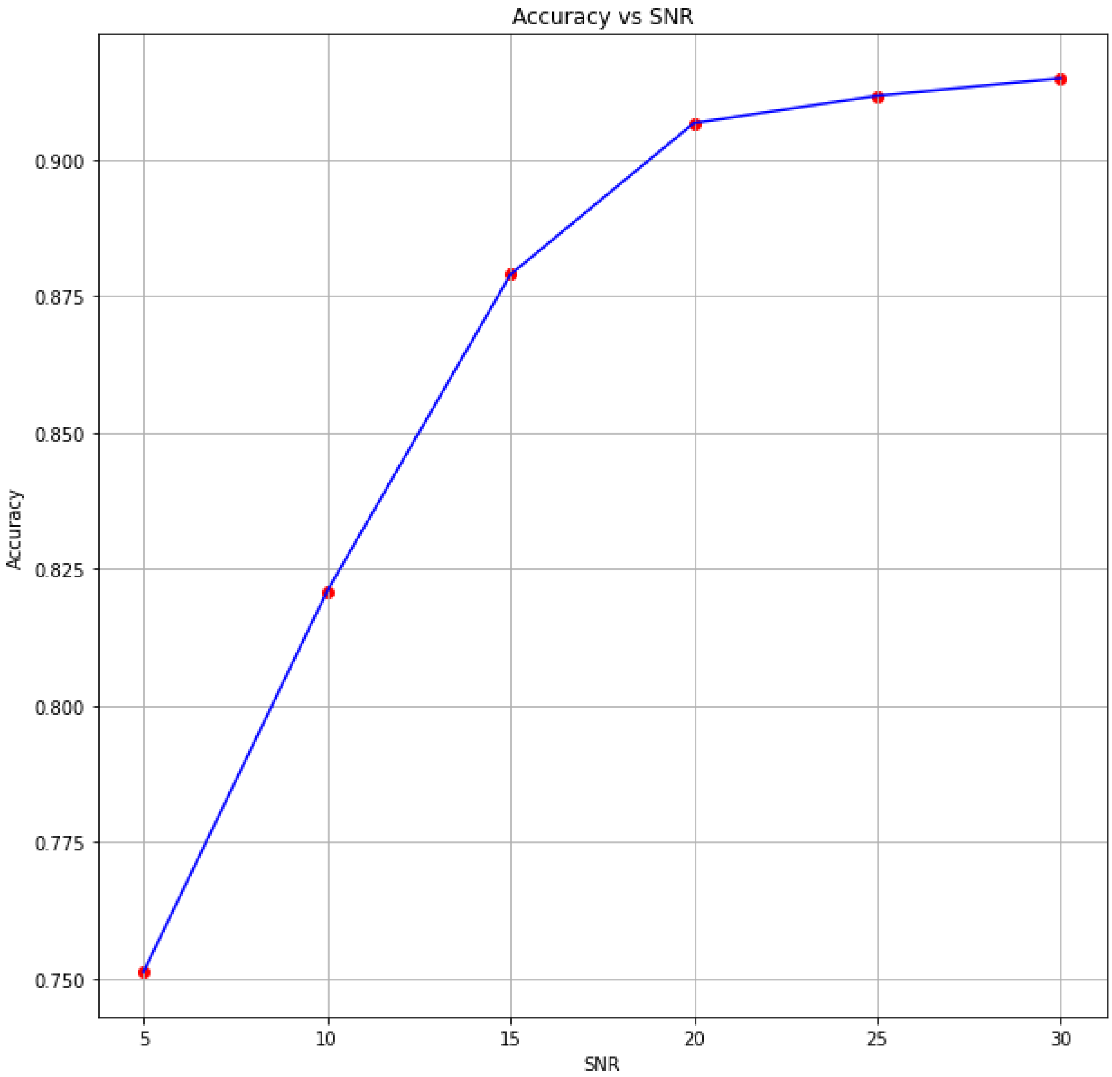
```
accuracy: 0.9066 - val_loss: 0.3343 - val_accuracy: 1.0000
Epoch 46/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2040 -
accuracy: 0.9065 - val_loss: 0.2917 - val_accuracy: 0.9997
Epoch 47/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2040 -
accuracy: 0.9064 - val_loss: 0.2954 - val_accuracy: 1.0000
Epoch 48/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2040 -
accuracy: 0.9065 - val_loss: 0.3125 - val_accuracy: 1.0000
Epoch 49/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2039 -
accuracy: 0.9065 - val_loss: 0.3090 - val_accuracy: 1.0000
Epoch 50/50
844/844 [==============================] - 2s 2ms/step - loss: 0.2040 -
accuracy: 0.9064 - val_loss: 0.3257 - val_accuracy: 0.9960
```

```
[9]: <tensorflow.python.keras.callbacks.History at 0x7fd9d43b6b50>
```

Evaluate the Model

```
[10]: EvaluateData(AWGN_Model, X_Train, y_Train, X_Valid, y_Valid, "AWGN_Accuracy.
       ↪jpg")
```

```
Evaluating Model
938/938 [==============================] - 2s 2ms/step - loss: 1.7904 -
accuracy: 0.7511
SNR: 5 Accuracy: 0.7511333227157593
938/938 [==============================] - 2s 2ms/step - loss: 1.0192 -
accuracy: 0.8208
SNR: 10 Accuracy: 0.8208333253860474
938/938 [==============================] - 2s 2ms/step - loss: 0.4816 -
accuracy: 0.8789
SNR: 15 Accuracy: 0.8788999915122986
938/938 [==============================] - 2s 2ms/step - loss: 0.2641 -
accuracy: 0.9067
SNR: 20 Accuracy: 0.9066666960716248
938/938 [==============================] - 2s 2ms/step - loss: 0.2244 -
accuracy: 0.9117
SNR: 25 Accuracy: 0.9116666913032532
938/938 [==============================] - 2s 2ms/step - loss: 0.2170 -
accuracy: 0.9149
SNR: 30 Accuracy: 0.9148666858673096
```



Save Model

```
[11]: AWGN_Model.save("Model/AWGN.h5")
```

1.6 AutoML Model for Rayleigh Channel of Channel-Length = 2

For Rayleigh Channel, Input Dimensions is (100,2)

Rayleigh Data

```
[12]: X_Train, y_Train, X_Valid, y_Valid = GenerateDatasets('Rayleigh',L=2)
```

Creating a CNN Classifier

[13]:

```
RayleighClassifier = ak.ImageClassifier(
    overwrite=True,
    max_trials=1,
    loss='categorical_crossentropy',
    num_classes=3)

RayleighClassifier.fit(X_Train, y_Train, epochs=5, batch_size=16,␣
 ↪validation_split=0.3)
```

```
Trial 1 Complete [00h 00m 16s]
val_loss: 0.11849616467952728

Best val_loss So Far: 0.11849616467952728
Total elapsed time: 00h 00m 16s
INFO:tensorflow:Oracle triggered exit
Epoch 1/5
938/938 [==============================] - 3s 3ms/step - loss: 0.1085 -
accuracy: 0.9769
Epoch 2/5
938/938 [==============================] - 3s 3ms/step - loss: 0.3147 -
accuracy: 0.8675
Epoch 3/5
938/938 [==============================] - 3s 3ms/step - loss: 0.2239 -
accuracy: 0.9413
Epoch 4/5
938/938 [==============================] - 3s 3ms/step - loss: 0.8871 -
accuracy: 0.9248
Epoch 5/5
938/938 [==============================] - 3s 3ms/step - loss: 0.1001 -
accuracy: 0.9685
INFO:tensorflow:Assets written to: ./image_classifier/best_model/assets
```

CNN Model Summary

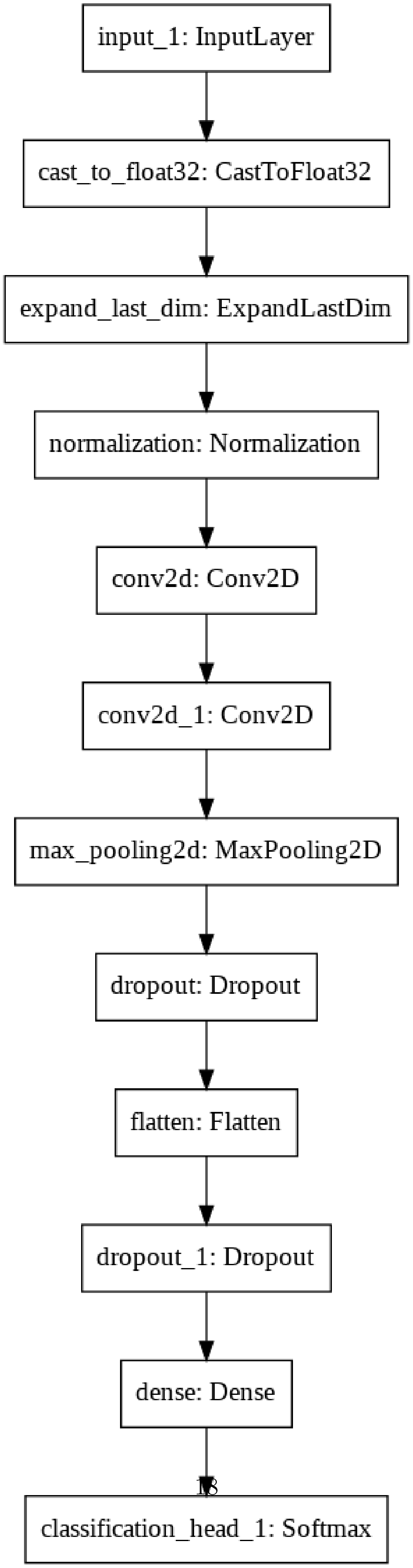
[14]:

```
Rayleigh_Model = RayleighClassifier.export_model()
Rayleigh_Model.summary()
tf.keras.utils.plot_model(Rayleigh_Model, to_file='Images/Rayleigh_Model_L=2.
 ↪png', show_shapes=False,show_layer_names=True)
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 100, 2)]          0
_________________________________________________________________
cast_to_float32 (CastToFloat (None, 100, 2)            0
_________________________________________________________________
```

```
expand_last_dim (ExpandLastD (None, 100, 2, 1)         0
_________________________________________________________________
normalization (Normalization (None, 100, 2, 1)         3
_________________________________________________________________
conv2d (Conv2D)              (None, 100, 2, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 100, 2, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 1, 64)         0
_________________________________________________________________
dropout (Dropout)            (None, 50, 1, 64)         0
_________________________________________________________________
flatten (Flatten)            (None, 3200)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 3200)              0
_________________________________________________________________
dense (Dense)                (None, 3)                 9603
_________________________________________________________________
classification_head_1 (Softm (None, 3)                 0
=================================================================
Total params: 28,422
Trainable params: 28,419
Non-trainable params: 3
_________________________________________________________________
```

[14]:


input_1: InputLayer
cast_to_float32: CastToFloat32
expand_last_dim: ExpandLastDim
normalization: Normalization
conv2d: Conv2D
conv2d_1: Conv2D
max_pooling2d: MaxPooling2D
dropout: Dropout
flatten: Flatten
dropout_1: Dropout
dense: Dense
classification_head_1: Softmax

1.6.1 Training and Evaluating CNN Model

Training the CNN Model

```
[15]: Rayleigh_Model.fit(X_Train, y_Train, epochs=20, batch_size=16,␣
       ↪validation_split=0.3,shuffle=True)
```

```
Epoch 1/20
657/657 [==============================] - 3s 4ms/step - loss: 0.1670 -
accuracy: 0.9650 - val_loss: 0.0312 - val_accuracy: 1.0000
Epoch 2/20
657/657 [==============================] - 2s 4ms/step - loss: 8.4203e-04 -
accuracy: 1.0000 - val_loss: 0.0037 - val_accuracy: 1.0000
Epoch 3/20
657/657 [==============================] - 2s 4ms/step - loss: 1.9418e-04 -
accuracy: 1.0000 - val_loss: 6.7732e-04 - val_accuracy: 1.0000
Epoch 4/20
657/657 [==============================] - 2s 4ms/step - loss: 3.2599e-05 -
accuracy: 1.0000 - val_loss: 1.4871e-04 - val_accuracy: 1.0000
Epoch 5/20
657/657 [==============================] - 2s 4ms/step - loss: 9.9750e-06 -
accuracy: 1.0000 - val_loss: 6.0170e-05 - val_accuracy: 1.0000
Epoch 6/20
657/657 [==============================] - 2s 4ms/step - loss: 5.3888e-06 -
accuracy: 1.0000 - val_loss: 2.6325e-05 - val_accuracy: 1.0000
Epoch 7/20
657/657 [==============================] - 2s 4ms/step - loss: 2.8481e-06 -
accuracy: 1.0000 - val_loss: 1.4253e-05 - val_accuracy: 1.0000
Epoch 8/20
657/657 [==============================] - 3s 4ms/step - loss: 1.1521e-06 -
accuracy: 1.0000 - val_loss: 7.4011e-06 - val_accuracy: 1.0000
Epoch 9/20
657/657 [==============================] - 3s 4ms/step - loss: 8.6505e-07 -
accuracy: 1.0000 - val_loss: 2.7228e-06 - val_accuracy: 1.0000
Epoch 10/20
657/657 [==============================] - 3s 4ms/step - loss: 4.0110e-07 -
accuracy: 1.0000 - val_loss: 1.0809e-06 - val_accuracy: 1.0000
Epoch 11/20
657/657 [==============================] - 3s 4ms/step - loss: 2.4870e-07 -
accuracy: 1.0000 - val_loss: 7.2877e-07 - val_accuracy: 1.0000
Epoch 12/20
657/657 [==============================] - 3s 4ms/step - loss: 9.6809e-08 -
accuracy: 1.0000 - val_loss: 3.0131e-07 - val_accuracy: 1.0000
Epoch 13/20
657/657 [==============================] - 2s 4ms/step - loss: 8.1628e-08 -
```

```
accuracy: 1.0000 - val_loss: 1.6660e-07 - val_accuracy: 1.0000
Epoch 14/20
657/657 [==============================] - 2s 4ms/step - loss: 6.5700e-08 -
accuracy: 1.0000 - val_loss: 1.1648e-07 - val_accuracy: 1.0000
Epoch 15/20
657/657 [==============================] - 2s 4ms/step - loss: 3.1255e-08 -
accuracy: 1.0000 - val_loss: 8.5513e-08 - val_accuracy: 1.0000
Epoch 16/20
657/657 [==============================] - 2s 4ms/step - loss: 2.6317e-08 -
accuracy: 1.0000 - val_loss: 6.7976e-08 - val_accuracy: 1.0000
Epoch 17/20
657/657 [==============================] - 3s 4ms/step - loss: 4.7647e-08 -
accuracy: 1.0000 - val_loss: 3.0973e-07 - val_accuracy: 1.0000
Epoch 18/20
657/657 [==============================] - 2s 4ms/step - loss: 2.8417e-08 -
accuracy: 1.0000 - val_loss: 1.2347e-07 - val_accuracy: 1.0000
Epoch 19/20
657/657 [==============================] - 2s 4ms/step - loss: 9.6162e-09 -
accuracy: 1.0000 - val_loss: 2.1723e-09 - val_accuracy: 1.0000
Epoch 20/20
657/657 [==============================] - 2s 4ms/step - loss: 1.5395e-08 -
accuracy: 1.0000 - val_loss: 7.9473e-11 - val_accuracy: 1.0000
```

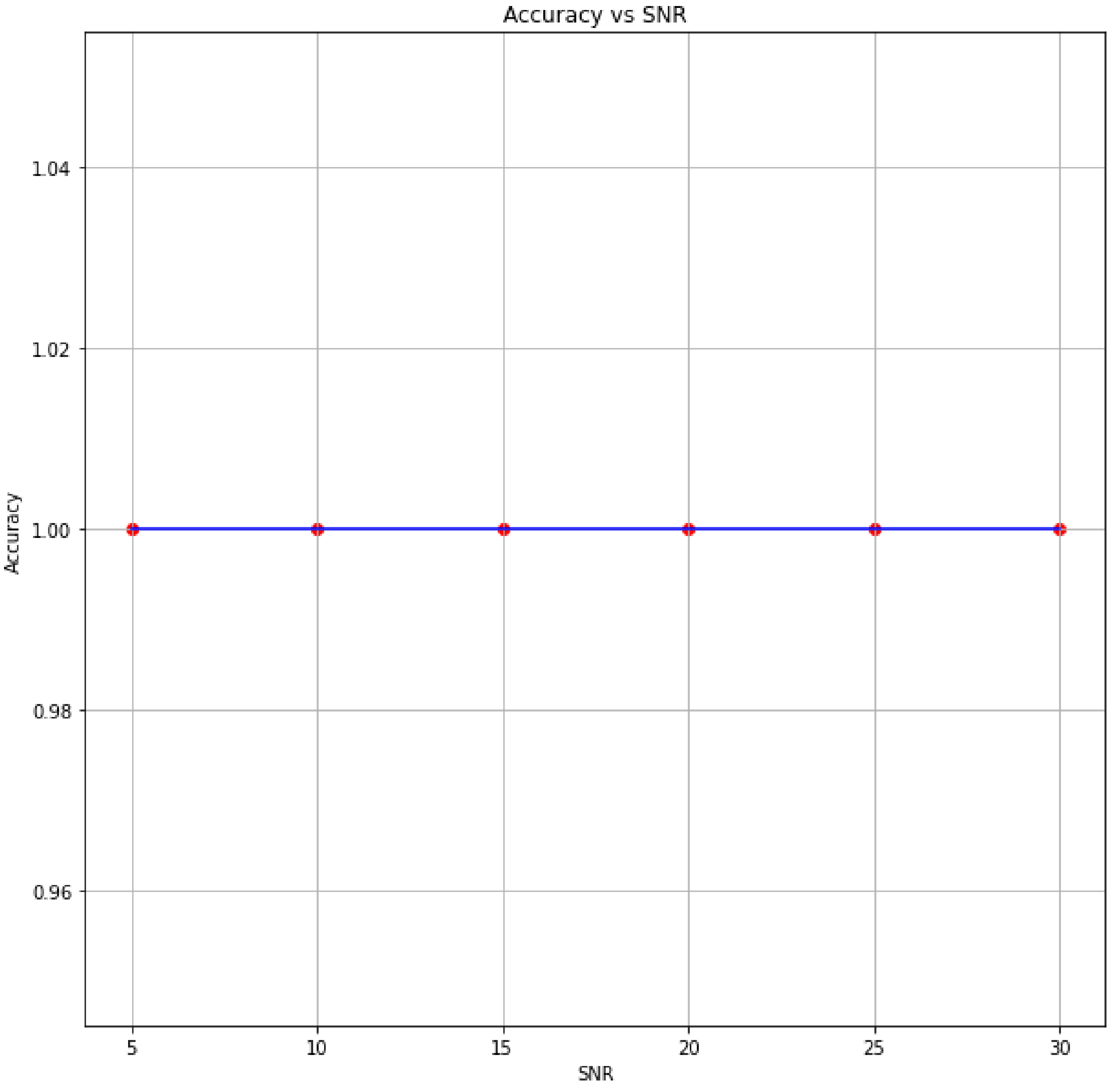
```
[15]: <tensorflow.python.keras.callbacks.History at 0x7fd9d464eb10>
```

Evaluate the CNN Model

```
[16]: EvaluateData(Rayleigh_Model, X_Train, y_Train, X_Valid, y_Valid,␣
       ↪"Rayleigh_Accuracy_L=2.jpg")
```

```
Evaluating Model
469/469 [==============================] - 1s 2ms/step - loss: 5.7395e-08 -
accuracy: 1.0000
SNR: 5 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 2.6385e-09 -
accuracy: 1.0000
SNR: 10 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 3.2584e-10 -
accuracy: 1.0000
SNR: 15 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 1.0331e-10 -
accuracy: 1.0000
SNR: 20 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 1.9868e-10 -
accuracy: 1.0000
SNR: 25 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 1.2716e-10 -
accuracy: 1.0000
```

```
SNR: 30 Accuracy: 1.0
```



Save CNN Model

```
[17]: Rayleigh_Model.save("Model/Rayleigh_L=2.h5")
```

1.7 AutoML Model for Rayleigh Channel of Channel-Length = 3

For Rayleigh Channel, Input Dimensions is (100,2)

Rayleigh Data

```
[18]: X_Train, y_Train, X_Valid, y_Valid = GenerateDatasets('Rayleigh',L=3)
```

Creating a CNN Classifier

[19]:
```
RayleighClassifier = ak.ImageClassifier(
    overwrite=True,
    max_trials=1,
    loss='categorical_crossentropy',
    num_classes=3)

RayleighClassifier.fit(X_Train, y_Train, epochs=5, batch_size=16,␣
 ↪validation_split=0.3)
```

```
Trial 1 Complete [00h 00m 14s]
val_loss: 0.1409987360239029

Best val_loss So Far: 0.1409987360239029
Total elapsed time: 00h 00m 14s
INFO:tensorflow:Oracle triggered exit
Epoch 1/5
938/938 [==============================] - 3s 3ms/step - loss: 0.0913 -
accuracy: 0.9736
Epoch 2/5
938/938 [==============================] - 3s 3ms/step - loss: 0.8848 -
accuracy: 0.9169
Epoch 3/5
938/938 [==============================] - 3s 3ms/step - loss: 0.4379 -
accuracy: 0.9280
Epoch 4/5
938/938 [==============================] - 3s 3ms/step - loss: 0.3910 -
accuracy: 0.9111
Epoch 5/5
938/938 [==============================] - 3s 3ms/step - loss: 0.3136 -
accuracy: 0.9426
INFO:tensorflow:Assets written to: ./image_classifier/best_model/assets
```

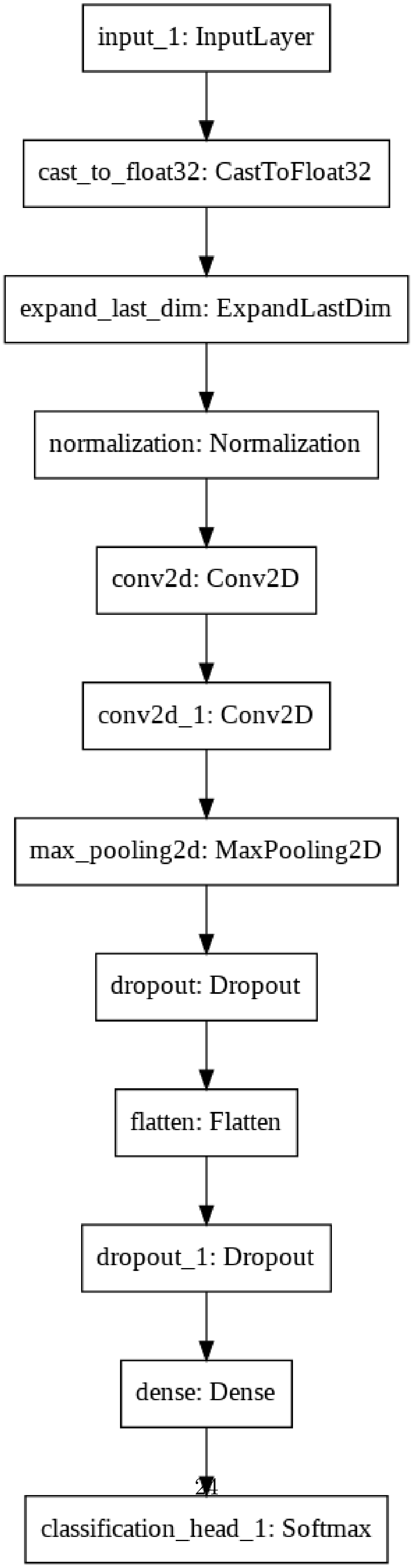
CNN Model Summary

[20]:
```
Rayleigh_Model = RayleighClassifier.export_model()
Rayleigh_Model.summary()
tf.keras.utils.plot_model(Rayleigh_Model, to_file='Images/Rayleigh_Model_L=3.
 ↪png', show_shapes=False,show_layer_names=True)
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 100, 2)]          0
_________________________________________________________________
cast_to_float32 (CastToFloat (None, 100, 2)            0
_________________________________________________________________
```

```
expand_last_dim (ExpandLastD (None, 100, 2, 1)         0

_________________________________________________________________
normalization (Normalization (None, 100, 2, 1)         3

_________________________________________________________________
conv2d (Conv2D)              (None, 100, 2, 32)        320

_________________________________________________________________
conv2d_1 (Conv2D)            (None, 100, 2, 64)        18496

_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 1, 64)         0

_________________________________________________________________
dropout (Dropout)            (None, 50, 1, 64)         0

_________________________________________________________________
flatten (Flatten)            (None, 3200)              0

_________________________________________________________________
dropout_1 (Dropout)          (None, 3200)              0

_________________________________________________________________
dense (Dense)                (None, 3)                 9603

_________________________________________________________________
classification_head_1 (Softm (None, 3)                 0
=================================================================
Total params: 28,422
Trainable params: 28,419
Non-trainable params: 3

_________________________________________________________________
```

[20]:


input_1: InputLayer
cast_to_float32: CastToFloat32
expand_last_dim: ExpandLastDim
normalization: Normalization
conv2d: Conv2D
conv2d_1: Conv2D
max_pooling2d: MaxPooling2D
dropout: Dropout
flatten: Flatten
dropout_1: Dropout
dense: Dense
classification_head_1: Softmax

1.7.1 Training and Evaluating CNN Model

Training the CNN Model

[21]:
```
Rayleigh_Model.fit(X_Train, y_Train, epochs=20, batch_size=16, validation_split=0.3,shuffle=True)
```

```
Epoch 1/20
657/657 [==============================] - 3s 4ms/step - loss: 0.1795 -
accuracy: 0.9681 - val_loss: 0.0129 - val_accuracy: 1.0000
Epoch 2/20
657/657 [==============================] - 2s 4ms/step - loss: 5.3318e-04 -
accuracy: 1.0000 - val_loss: 0.0014 - val_accuracy: 1.0000
Epoch 3/20
657/657 [==============================] - 2s 4ms/step - loss: 8.8514e-05 -
accuracy: 1.0000 - val_loss: 3.9895e-04 - val_accuracy: 1.0000
Epoch 4/20
657/657 [==============================] - 3s 4ms/step - loss: 3.5372e-05 -
accuracy: 1.0000 - val_loss: 1.4485e-04 - val_accuracy: 1.0000
Epoch 5/20
657/657 [==============================] - 2s 4ms/step - loss: 1.6374e-05 -
accuracy: 1.0000 - val_loss: 7.3171e-05 - val_accuracy: 1.0000
Epoch 6/20
657/657 [==============================] - 2s 4ms/step - loss: 7.3211e-06 -
accuracy: 1.0000 - val_loss: 3.4351e-05 - val_accuracy: 1.0000
Epoch 7/20
657/657 [==============================] - 2s 4ms/step - loss: 5.0201e-06 -
accuracy: 1.0000 - val_loss: 1.9795e-05 - val_accuracy: 1.0000
Epoch 8/20
657/657 [==============================] - 2s 4ms/step - loss: 3.7358e-06 -
accuracy: 1.0000 - val_loss: 1.1241e-05 - val_accuracy: 1.0000
Epoch 9/20
657/657 [==============================] - 2s 4ms/step - loss: 2.0092e-06 -
accuracy: 1.0000 - val_loss: 6.8868e-06 - val_accuracy: 1.0000
Epoch 10/20
657/657 [==============================] - 2s 4ms/step - loss: 1.4446e-06 -
accuracy: 1.0000 - val_loss: 3.8716e-06 - val_accuracy: 1.0000
Epoch 11/20
657/657 [==============================] - 2s 4ms/step - loss: 7.6345e-07 -
accuracy: 1.0000 - val_loss: 2.5221e-06 - val_accuracy: 1.0000
Epoch 12/20
657/657 [==============================] - 2s 4ms/step - loss: 8.3578e-07 -
accuracy: 1.0000 - val_loss: 1.5626e-06 - val_accuracy: 1.0000
Epoch 13/20
657/657 [==============================] - 2s 4ms/step - loss: 3.8577e-07 -
```

```
accuracy: 1.0000 - val_loss: 9.9079e-07 - val_accuracy: 1.0000
Epoch 14/20
657/657 [==============================] - 3s 4ms/step - loss: 2.4486e-07 -
accuracy: 1.0000 - val_loss: 6.0670e-07 - val_accuracy: 1.0000
Epoch 15/20
657/657 [==============================] - 2s 4ms/step - loss: 2.1856e-07 -
accuracy: 1.0000 - val_loss: 4.6168e-07 - val_accuracy: 1.0000
Epoch 16/20
657/657 [==============================] - 2s 4ms/step - loss: 1.4611e-07 -
accuracy: 1.0000 - val_loss: 2.3855e-07 - val_accuracy: 1.0000
Epoch 17/20
657/657 [==============================] - 2s 4ms/step - loss: 1.4674e-07 -
accuracy: 1.0000 - val_loss: 2.4160e-07 - val_accuracy: 1.0000
Epoch 18/20
657/657 [==============================] - 2s 4ms/step - loss: 1.0978e-07 -
accuracy: 1.0000 - val_loss: 1.1921e-07 - val_accuracy: 1.0000
Epoch 19/20
657/657 [==============================] - 2s 4ms/step - loss: 3.9614e-07 -
accuracy: 1.0000 - val_loss: 3.2425e-07 - val_accuracy: 1.0000
Epoch 20/20
657/657 [==============================] - 2s 4ms/step - loss: 7.2274e-08 -
accuracy: 1.0000 - val_loss: 1.0724e-07 - val_accuracy: 1.0000
```

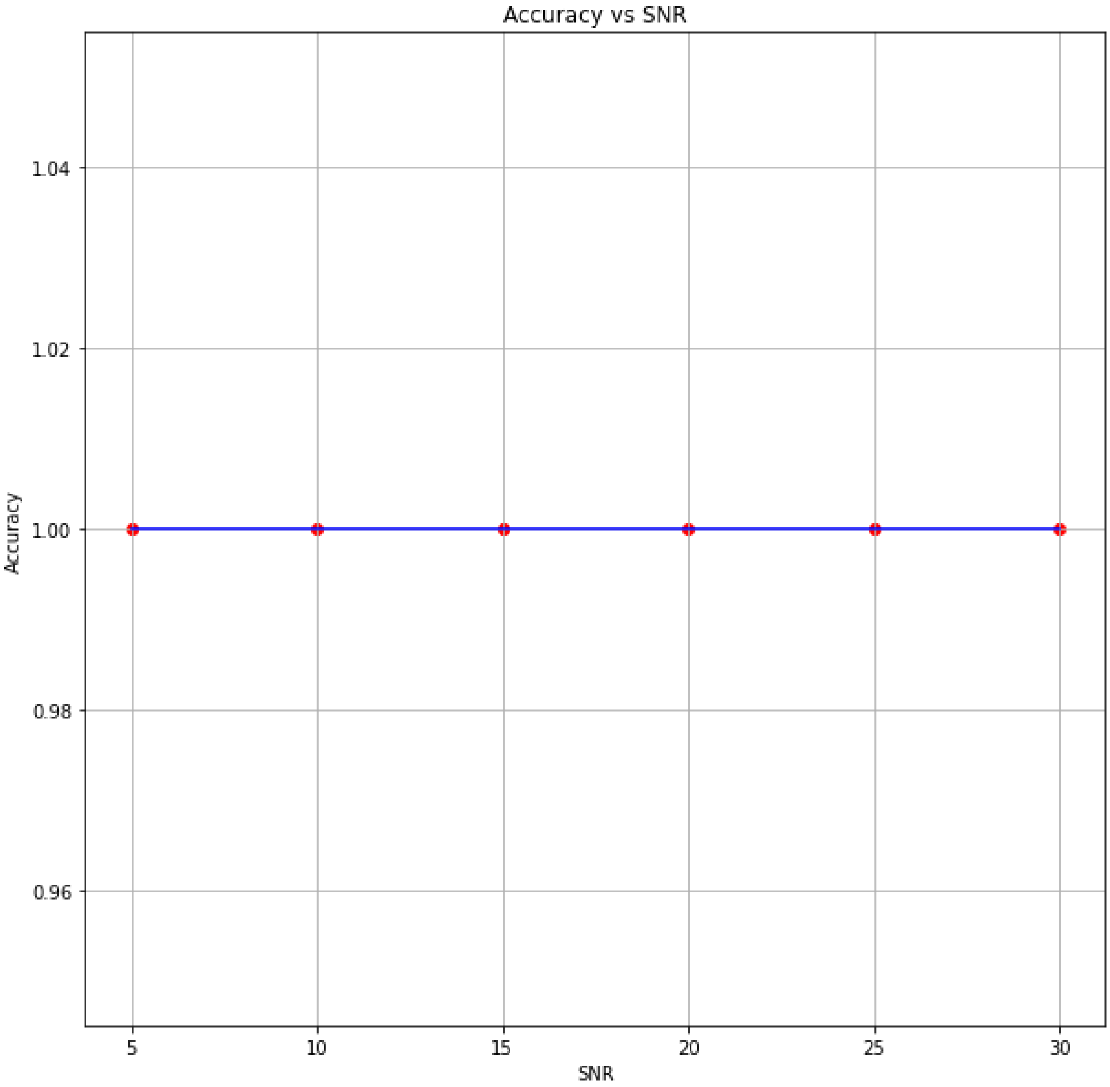
```
[21]: <tensorflow.python.keras.callbacks.History at 0x7fda2011c110>
```

Evaluate the CNN Model

```
[22]: EvaluateData(Rayleigh_Model, X_Train, y_Train, X_Valid, y_Valid,␣
       ↪"Rayleigh_Accuracy_L=3.jpg")
```

```
Evaluating Model
469/469 [==============================] - 1s 2ms/step - loss: 4.9734e-08 -
accuracy: 1.0000
SNR: 5 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 3.9212e-08 -
accuracy: 1.0000
SNR: 10 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 3.7503e-08 -
accuracy: 1.0000
SNR: 15 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 3.6589e-08 -
accuracy: 1.0000
SNR: 20 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 3.5834e-08 -
accuracy: 1.0000
SNR: 25 Accuracy: 1.0
469/469 [==============================] - 1s 2ms/step - loss: 3.5779e-08 -
accuracy: 1.0000
```

```
SNR: 30 Accuracy: 1.0
```



Save CNN Model

[23]:
```
Rayleigh_Model.save("Model/Rayleigh_L=3.h5")
```