Deep Learning for Automatic Modulation Classification

Under Dr. Kamal Captain:

Group 34

- Aditya Saggar (U17EC048)
- Devang Amin (U17EC123)
- Lavanya Pareek (U17EC093)

Presentation Timeline

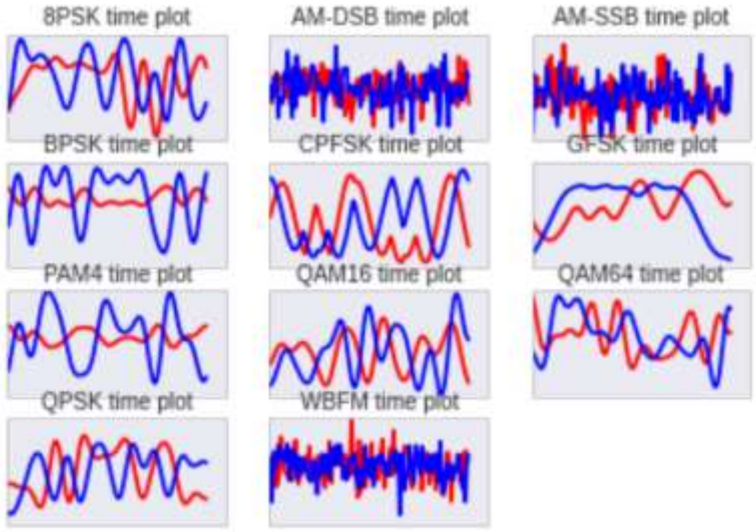
- Introduction to Modulation Classification
- Current Limitations
- Motivation to use Deep Learning
- Problem Statement & Objective
- Dataset Evaluation
- Neural Network
- Results and Future Scope

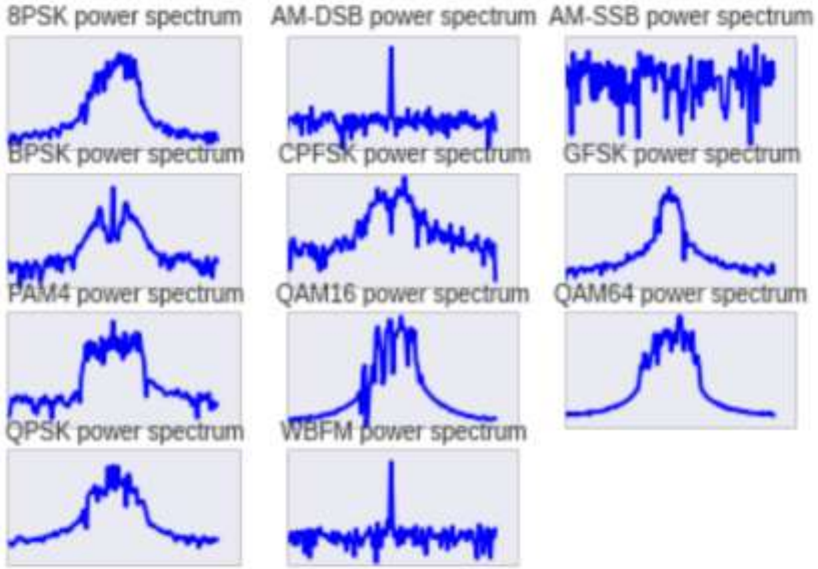
Modulation Recognition: Introduction

"Task of classifying the modulation type of a received radio signal as a step towards understanding what type of communications scheme and emitter is present"

- Civilian Applications
- Military Applications
- Cognitive Radio: Dynamic Spectrum Access

Modulation Recognition: Introduction





Modulation Recognition: Introduction

Received signal can be written as:

$$y(t)=f(s(t))+n(t)$$

11 Different Modulation Schemes :

BPSK, QPSK, 8PSK, 16QAM, 64QAM, BFSK, CPFSK, PAM4, WB-FM, AM-SSB, AND AM-DSB.

Existing Methods:

- Decision Theory Based Method
- Statistical Pattern Recognition Based Method

Modulation Recognition: Current Limitations

We require prior information of received signals. For large amount of modulation types :

- Time consuming
- Inefficient
- Lack of flexibility
- Feature design difficulties for non cooperative environment
- Likelihood systems not ideal for unknown channel conditions
- Heavy computational load

Motivation to use Deep Learning

- Can be seen as a pattern recognition problem, recognition model can be learned with multiple unknown parameters based on sufficiently large training data.
- Success in video, speech, image recognition problems
- Improved research accessibility
- Better demonstrated performance than feature based systems
- To add robustness, flexibility and accuracy

Problem Statement

" N class decision problem where we input a complex time series signal (with in- phase and quadrature components) and train a network on windowed sequences of the time series signal. "

Objective

" To add robustness and accuracy to modulation classification at the receiving end of transmission using deep learning techniques to accomplish that and finally compare different techniques used "

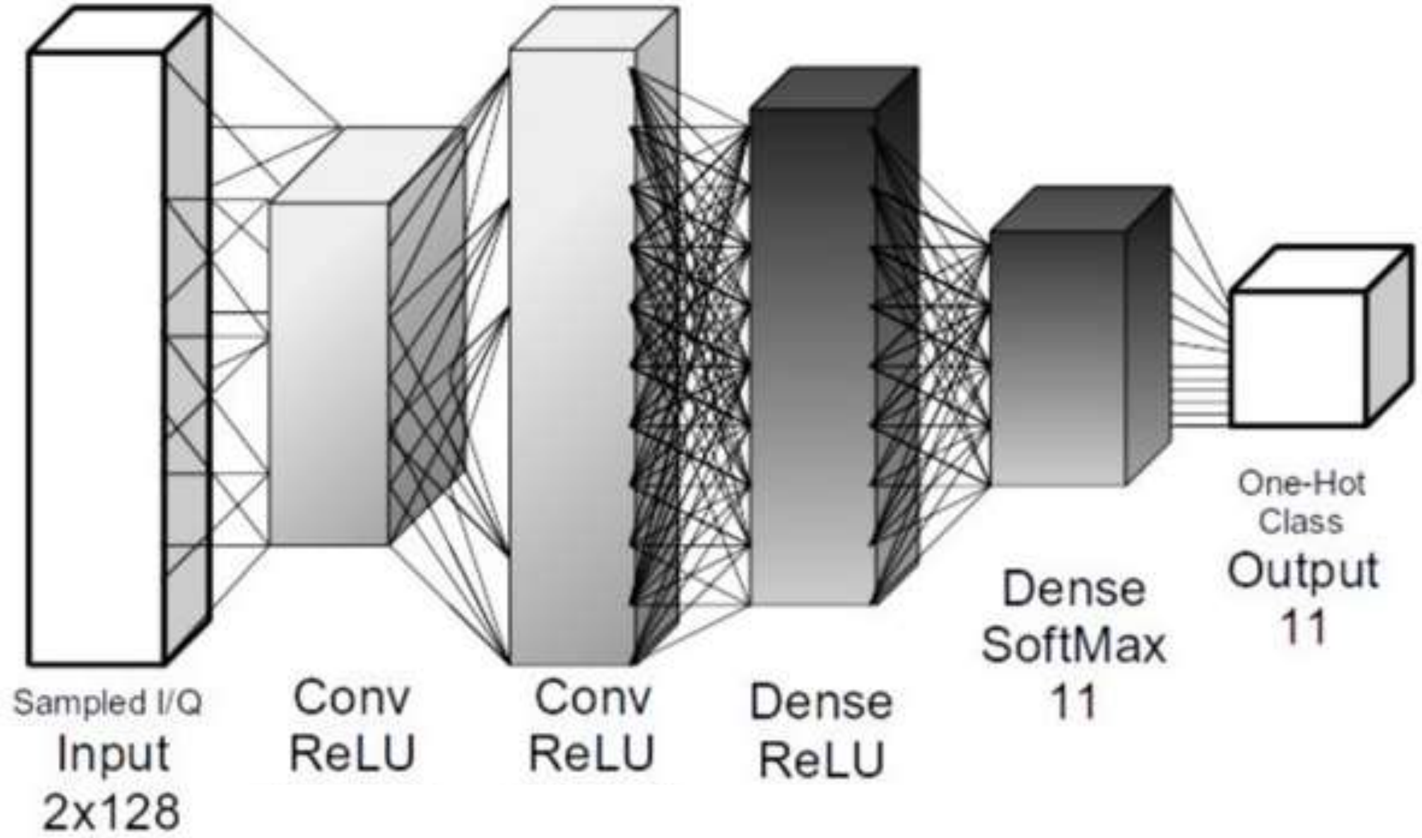
Dataset

Dataset	RadioML2016.10a
Modulations	8 Digital Modulations: BPSK, QPSK, 8PSK, 16QAM, 64QAM, BFSK, CPFSK, and PAM4 3 Analog Modulations: WBFM, AM-SSB, and AM-DSB
Length per sample	128
Signal format	In-phase and quadrature (IQ)
Signal dimension	2×128 per Sample
Duration per sample	128 μs
Sampling frequency	1 MHz
Samples per symbol	8
SNR Range	[-20 dB, -18 dB, -16 dB, . . ., 18 dB]
Total number of samples	220000 vectors

Networks Deployed

- Convolutional Neural Network
- Long Short Term Memory
- CNN- LSTM
- ResNet
- VGG
- InceptionNet
- Dual Stream CNN/ CNN-LSTM
- Attention Modules

Convolutional Neural Network : General Architecture

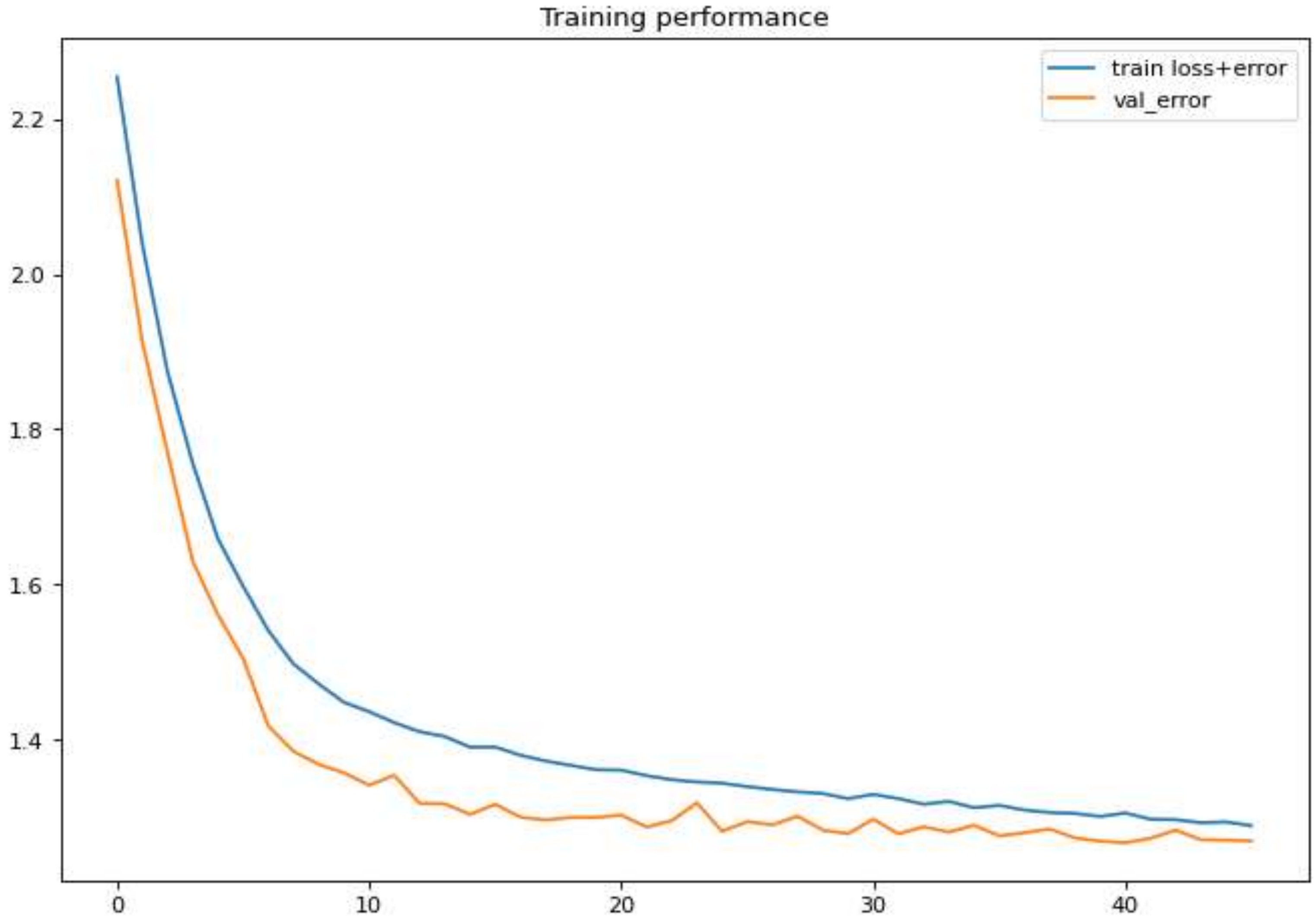


[O'Shea, Timothy J., Johnathan Corgan, and T. Charles Clancy. "Convolutional radio modulation recognition networks." *International conference on engineering applications of neural networks*. Springer, Cham, 2016.]

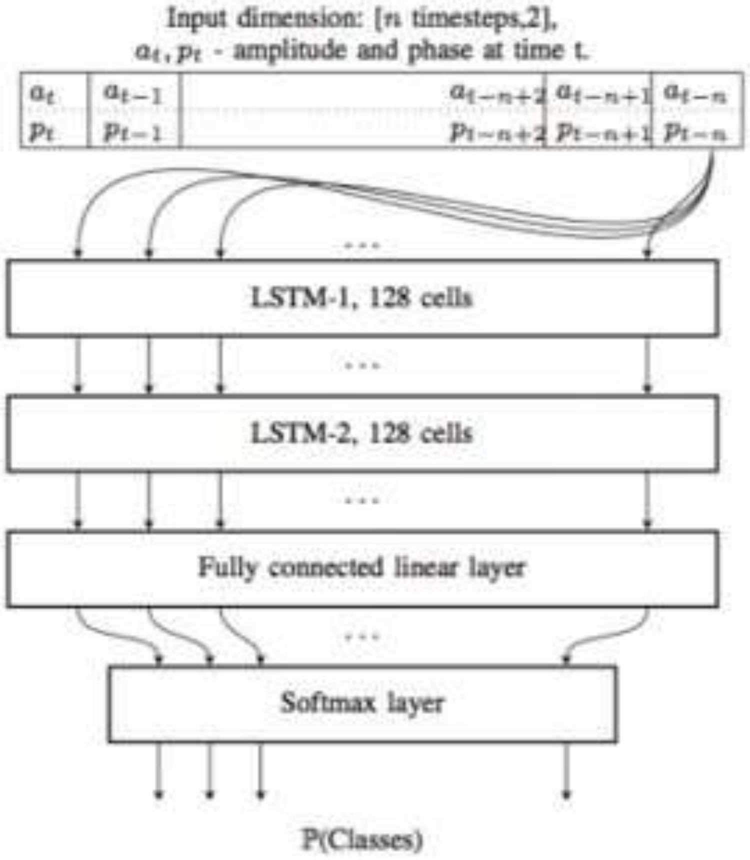
Convolutional Neural Network : Implemented Sequential Model

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 2, 128, 1)         0
_________________________________________________________________
zero_padding2d (ZeroPadding2 (None, 2, 132, 1)         0
_________________________________________________________________
conv2d (Conv2D)              (None, 2, 130, 256)       1024
_________________________________________________________________
dropout (Dropout)            (None, 2, 130, 256)       0
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 2, 134, 256)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 1, 132, 80)        122960
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 132, 80)        0
_________________________________________________________________
flatten (Flatten)            (None, 10560)             0
_________________________________________________________________
dense1 (Dense)               (None, 256)               2703616
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense2 (Dense)               (None, 11)                2827
_________________________________________________________________
activation (Activation)      (None, 11)                0
_________________________________________________________________
reshape_1 (Reshape)          (None, 11)                0
=================================================================
Total params: 2,830,427
Trainable params: 2,830,427
Non-trainable params: 0
_________________________________________________________________
```

Convolutional Neural Network : Training Performance



Long Short Term Memory: General Architecture



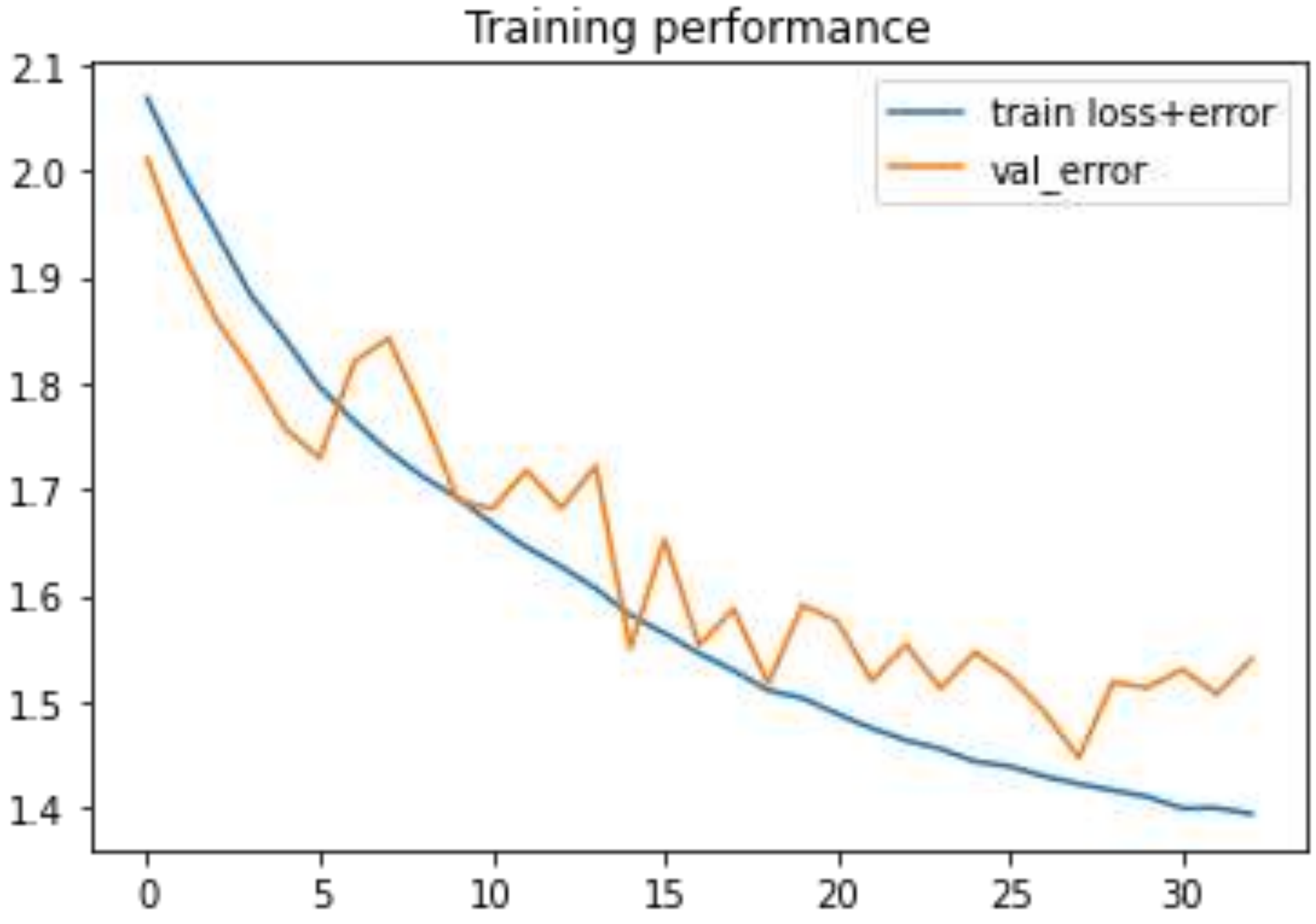
[Ramjee, Sharan, et al. "Fast deep learning for automatic modulation classification." *arXiv preprint arXiv:1901.05850* (2019).]

Long Short Term Memory: Implemented Sequential Model

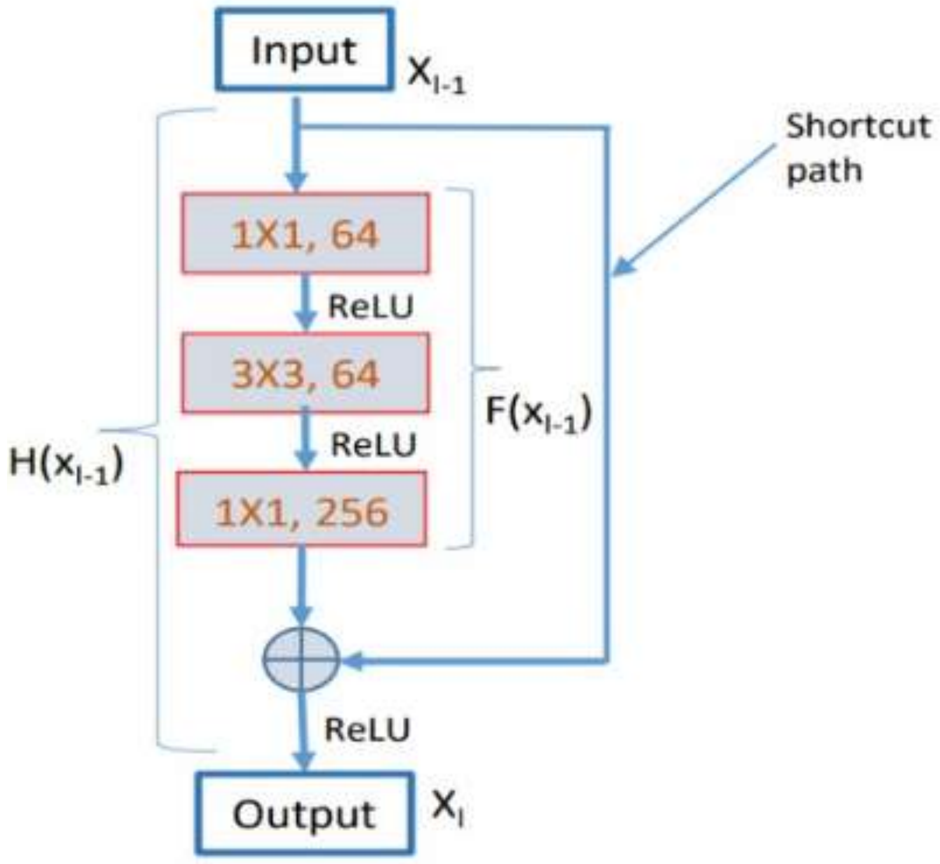
Layer (type)	Output Shape	Param #
lstm1 (LSTM)	(None, 128, 128)	67072
lstm2 (LSTM)	(None, 128, 128)	131584
flatten_1 (Flatten)	(None, 16384)	0
dense (Dense)	(None, 11)	180235

Total params: 378,891
Trainable params: 378,891
Non-trainable params: 0

Long Short Term Memory: Training Performance



ResNet: General Architecture

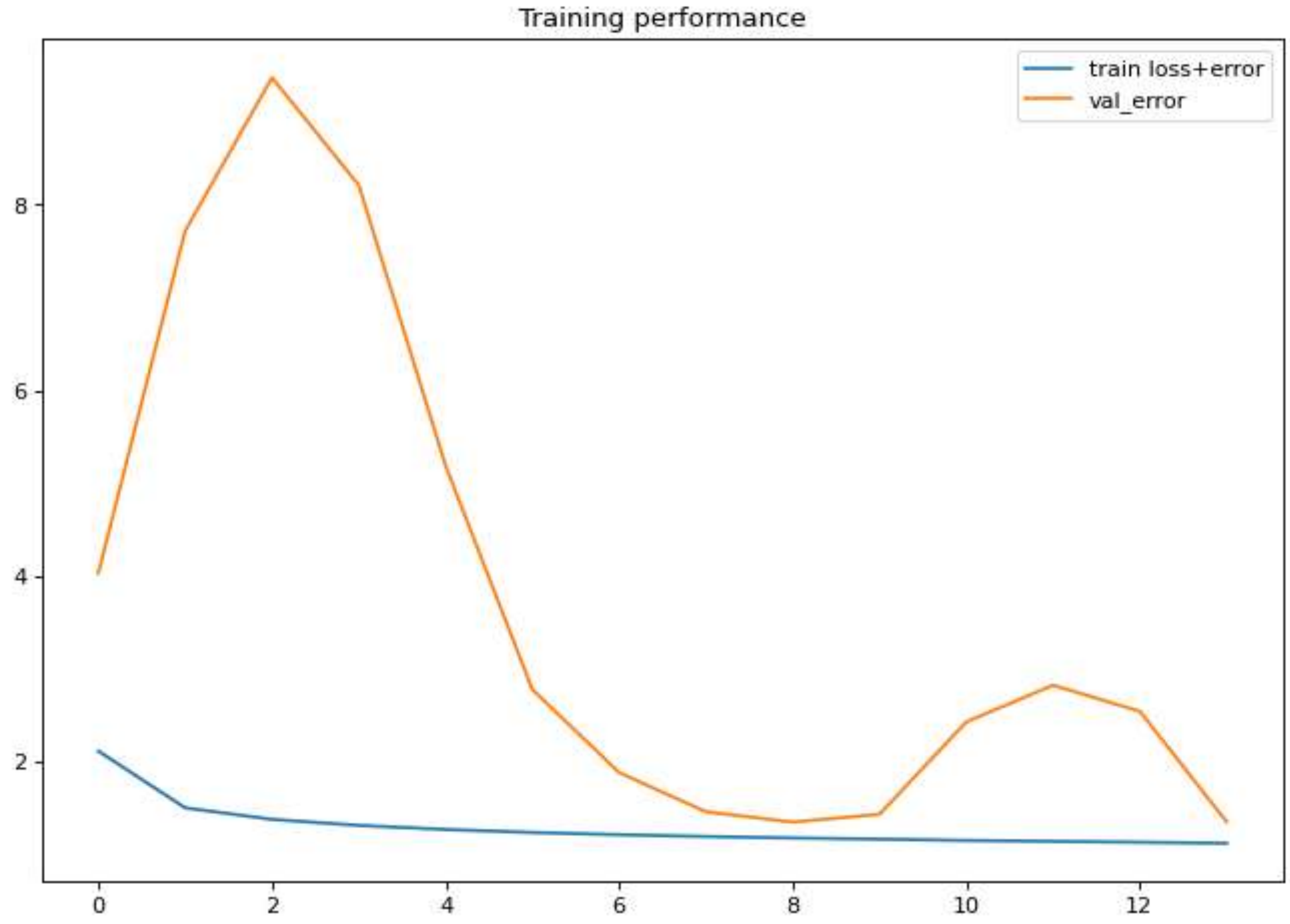


[Kumar, Yogesh, et al. "Automatic modulation classification based on constellation density using deep learning." IEEE Communications Letters 24.6 (2020): 1275-1278.]

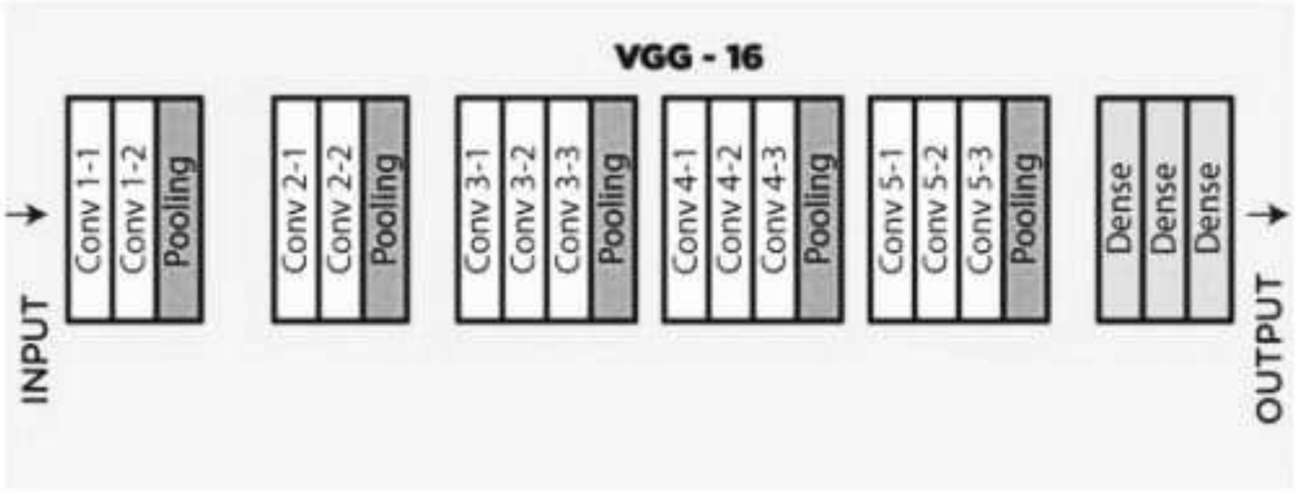
ResNet: Implemented Model

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape_11 (Reshape)         (None, 2, 128, 1)         0
_________________________________________________________________
model_10 (Functional)        (None, 11)                54603
=================================================================
Total params: 54,603
Trainable params: 52,811
Non-trainable params: 1,792
_________________________________________________________________
```

ResNet: Training Performance



VGG: General Architecture

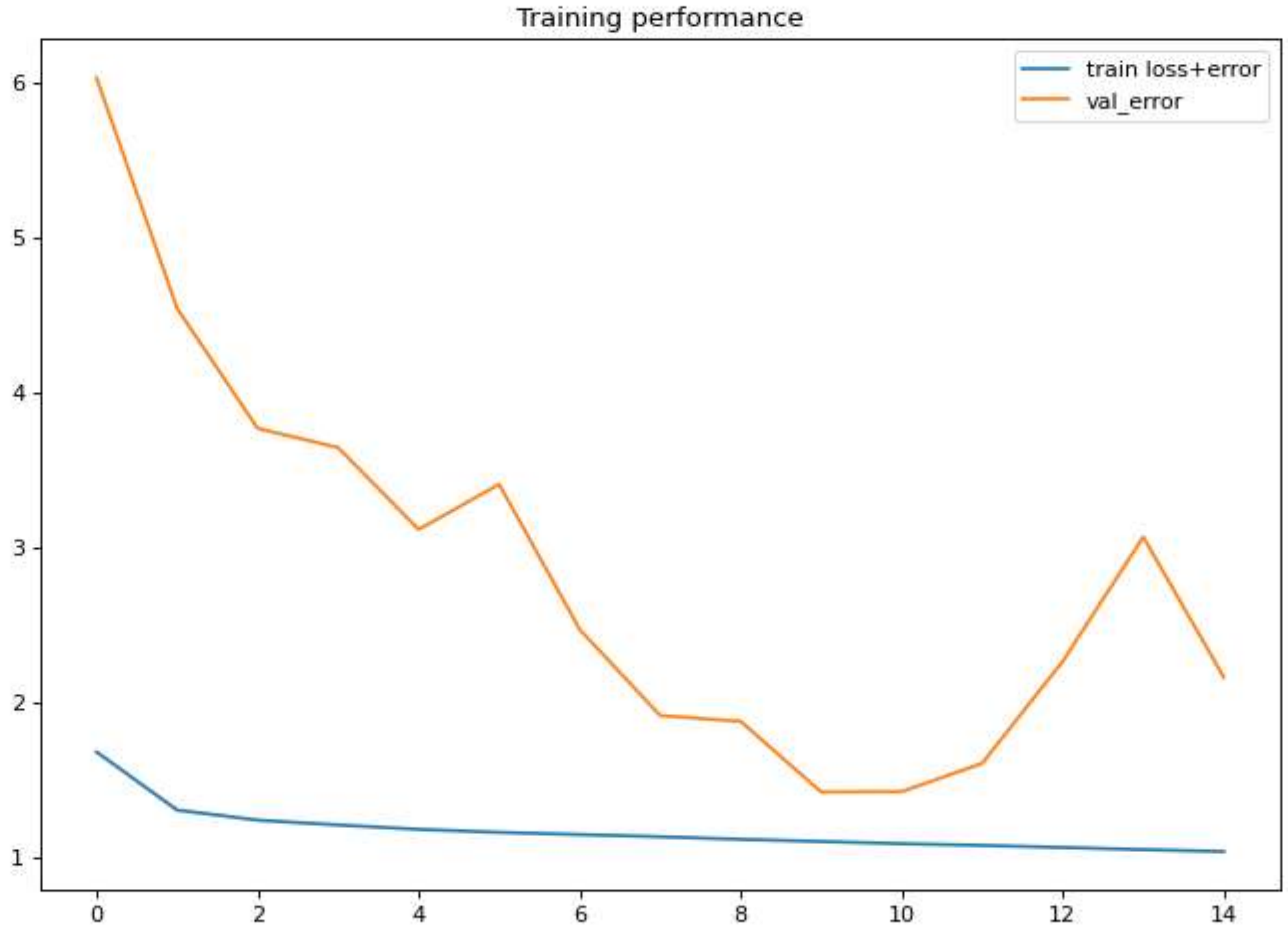


[“VGG-16: CNN Model.” *GeeksforGeeks*, 27 Feb. 2020, www.geeksforgeeks.org/vgg-16-cnn-model/.]

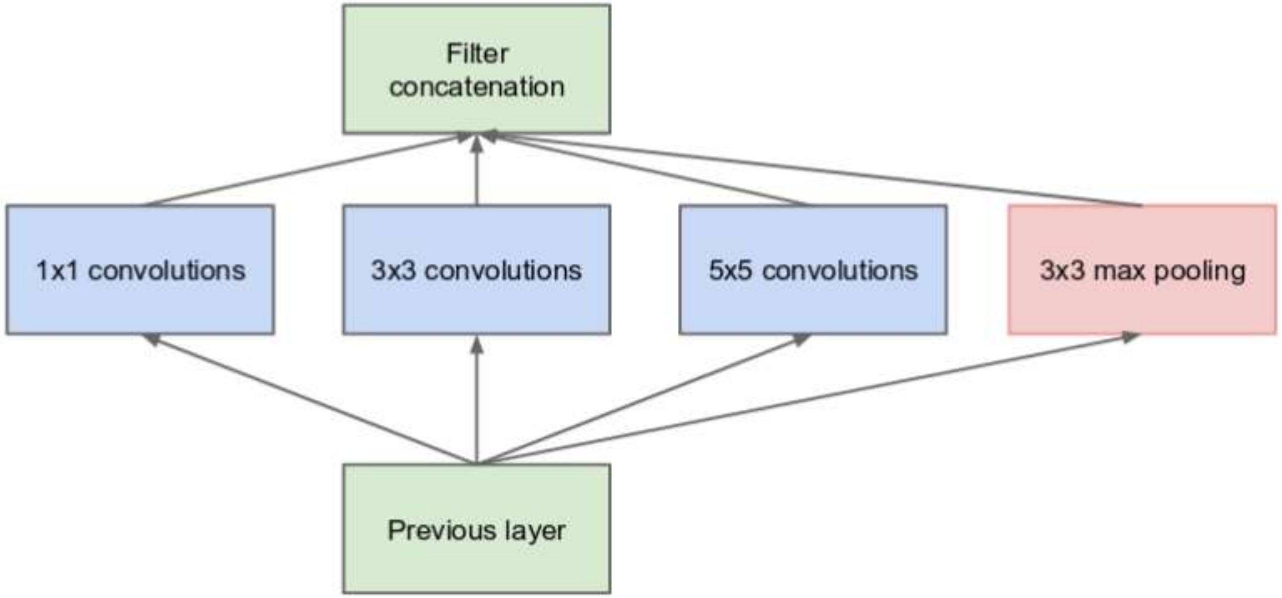
VGG: Implemented Sequential Model

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape_10 (Reshape)         (None, 2, 128, 1)         0
_________________________________________________________________
model_9 (Functional)         (None, 11)                177867
=================================================================
Total params: 177,867
Trainable params: 176,971
Non-trainable params: 896
_________________________________________________________________
```

VGG: Training Performance



InceptionNet: General Architecture



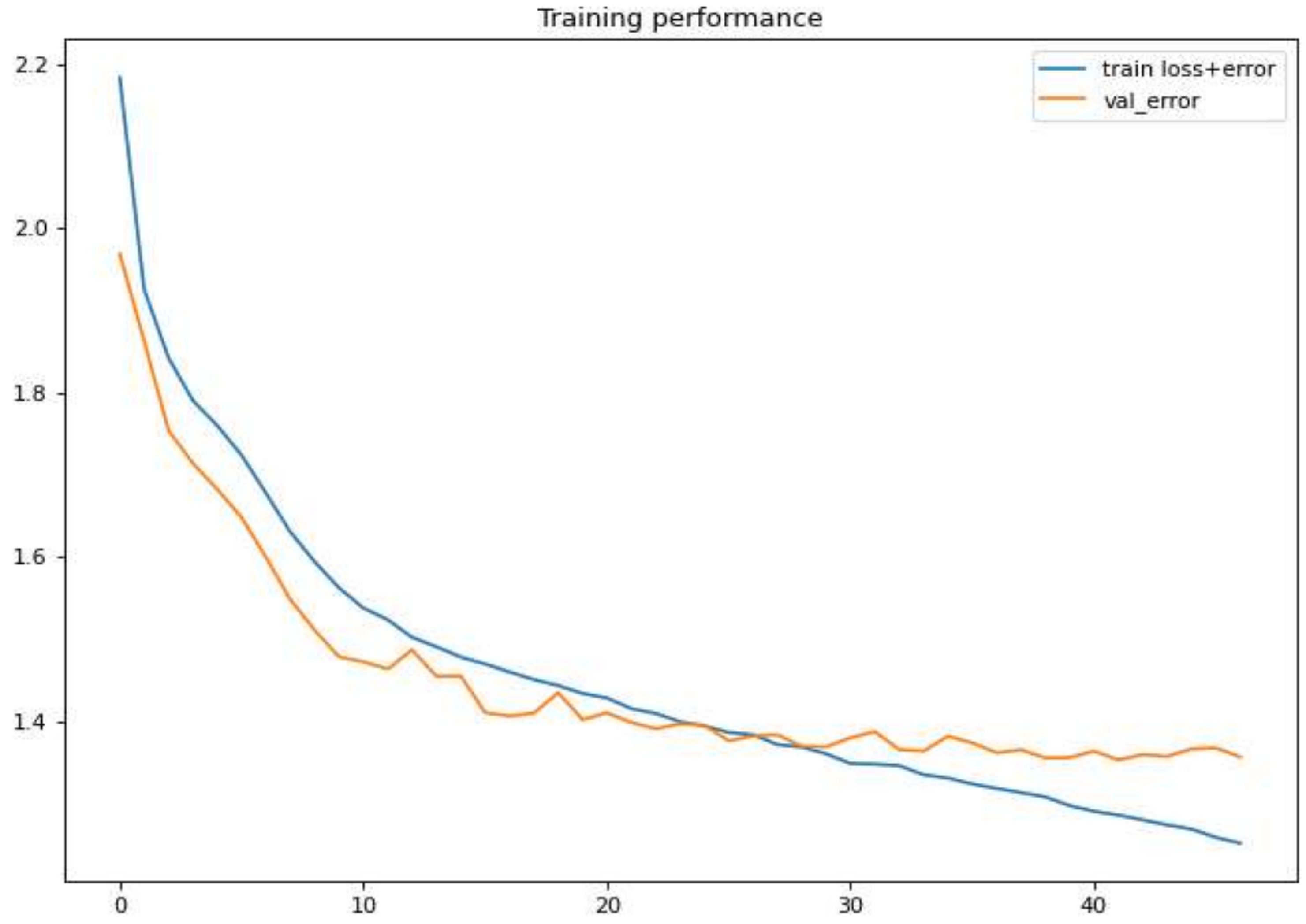
[Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014.]

InceptionNet: Implemented Model

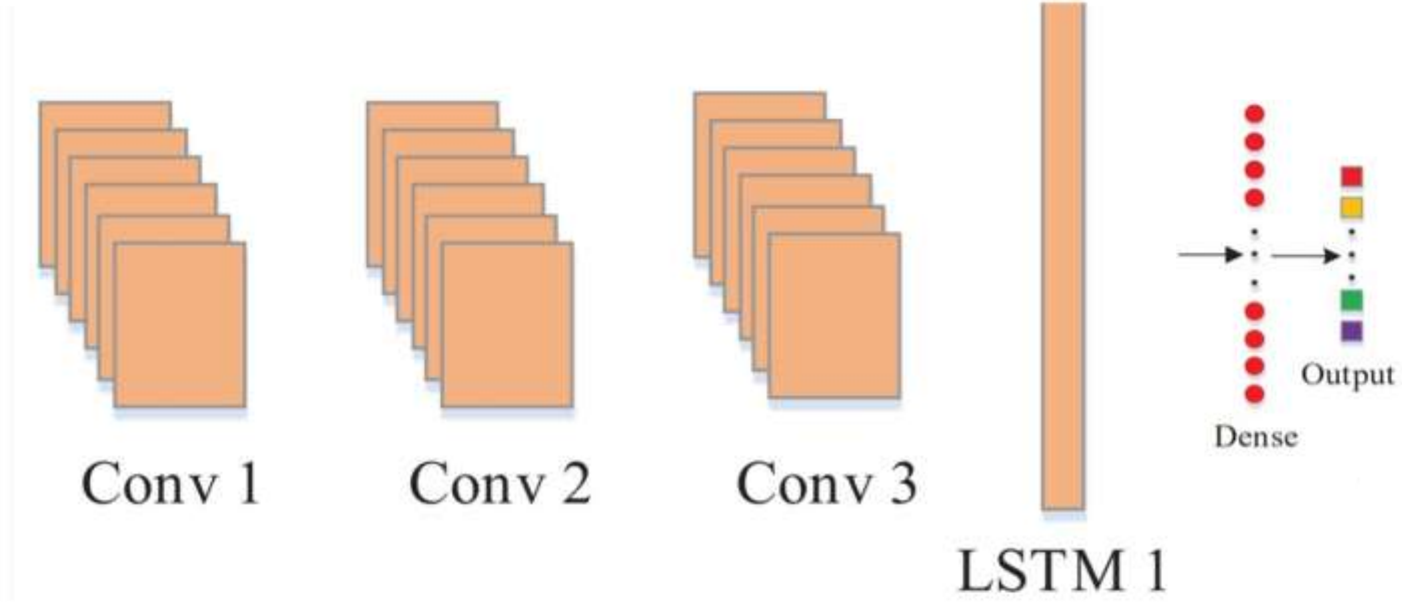
Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 1, 2, 128)]	0	
zero_padding2d_3 (ZeroPadding2D	(None, 1, 2, 132)	0	input_2[0][0]
conv11 (Conv2D)	(None, 50, 2, 132)	100	zero_padding2d_3[0][0]
conv21 (Conv2D)	(None, 50, 2, 132)	100	zero_padding2d_3[0][0]
dropout_6 (Dropout)	(None, 50, 2, 132)	0	conv11[0][0]
dropout_8 (Dropout)	(None, 50, 2, 132)	0	conv21[0][0]
zero_padding2d_4 (ZeroPadding2D	(None, 50, 2, 136)	0	dropout_6[0][0]
zero_padding2d_5 (ZeroPadding2D	(None, 50, 2, 136)	0	dropout_8[0][0]
conv12 (Conv2D)	(None, 50, 2, 129)	20050	zero_padding2d_4[0][0]
conv22 (Conv2D)	(None, 50, 2, 134)	7550	zero_padding2d_5[0][0]
conv31 (Conv2D)	(None, 50, 2, 132)	100	zero_padding2d_3[0][0]
dropout_7 (Dropout)	(None, 50, 2, 129)	0	conv12[0][0]
dropout_9 (Dropout)	(None, 50, 2, 134)	0	conv22[0][0]
dropout_10 (Dropout)	(None, 50, 2, 132)	0	conv31[0][0]
concatenate_1 (Concatenate)	(None, 50, 2, 395)	0	dropout_7[0][0] dropout_9[0][0] dropout_10[0][0]
flatten_1 (Flatten)	(None, 39500)	0	concatenate_1[0][0]
dense1 (Dense)	(None, 256)	10112256	flatten_1[0][0]
dropout_11 (Dropout)	(None, 256)	0	dense1[0][0]
dense2 (Dense)	(None, 11)	2827	dropout_11[0][0]
activation_1 (Activation)	(None, 11)	0	dense2[0][0]
reshape_1 (Reshape)	(None, 11)	0	activation_1[0][0]

Total params: 10,142,983
Trainable params: 10,142,983
Non-trainable params: 0

InceptionNet: Training Performance



CNN-LSTM: General Architecture



Zhang, Zufan, et al. "Automatic Modulation Classification Using CNN-LSTM Based Dual-Stream Structure." *IEEE Transactions on Vehicular Technology* 69.11 (2020): 13521-13531.

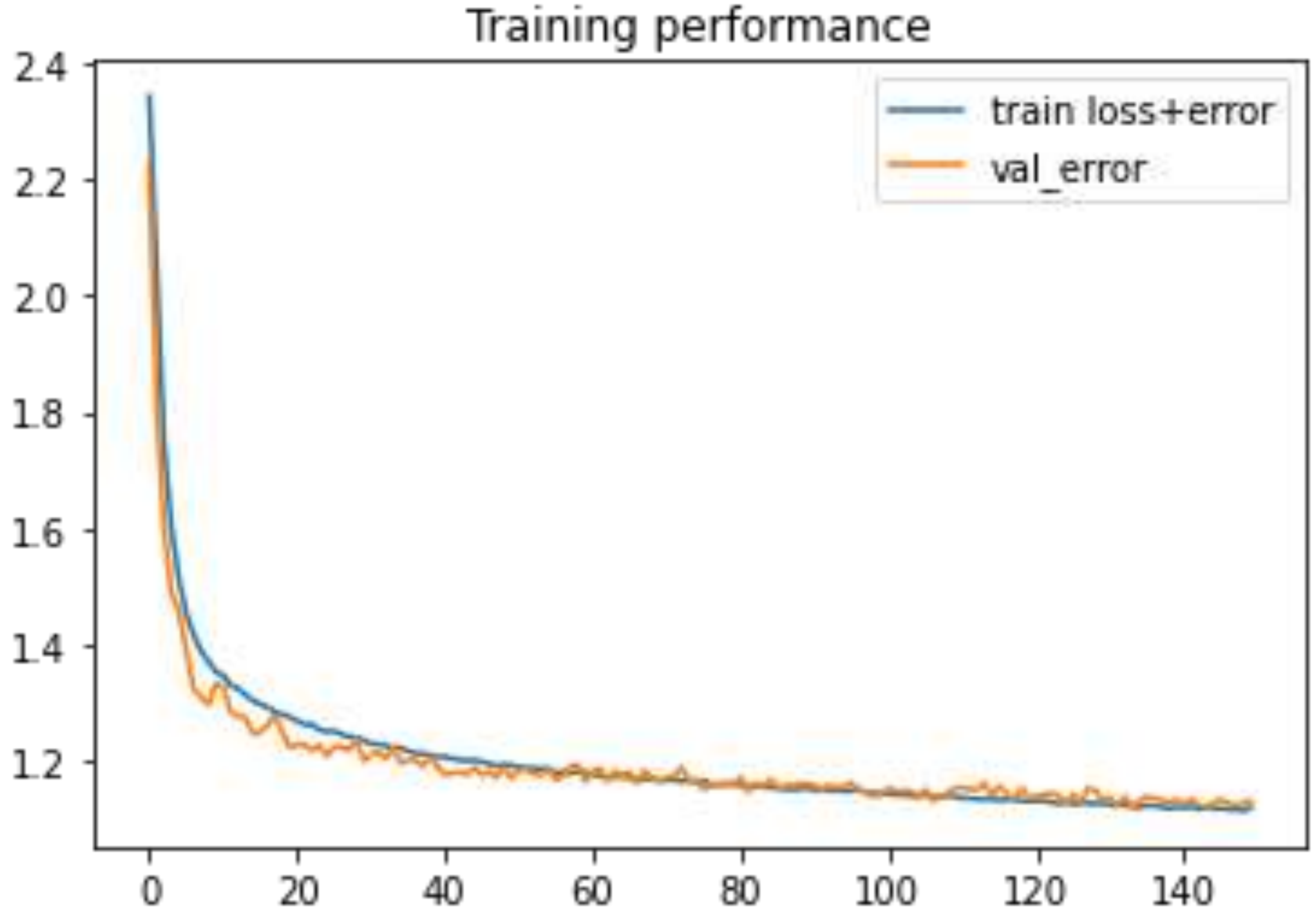
CNN-LSTM: Implemented Sequential Model

Model: "sequential_10"

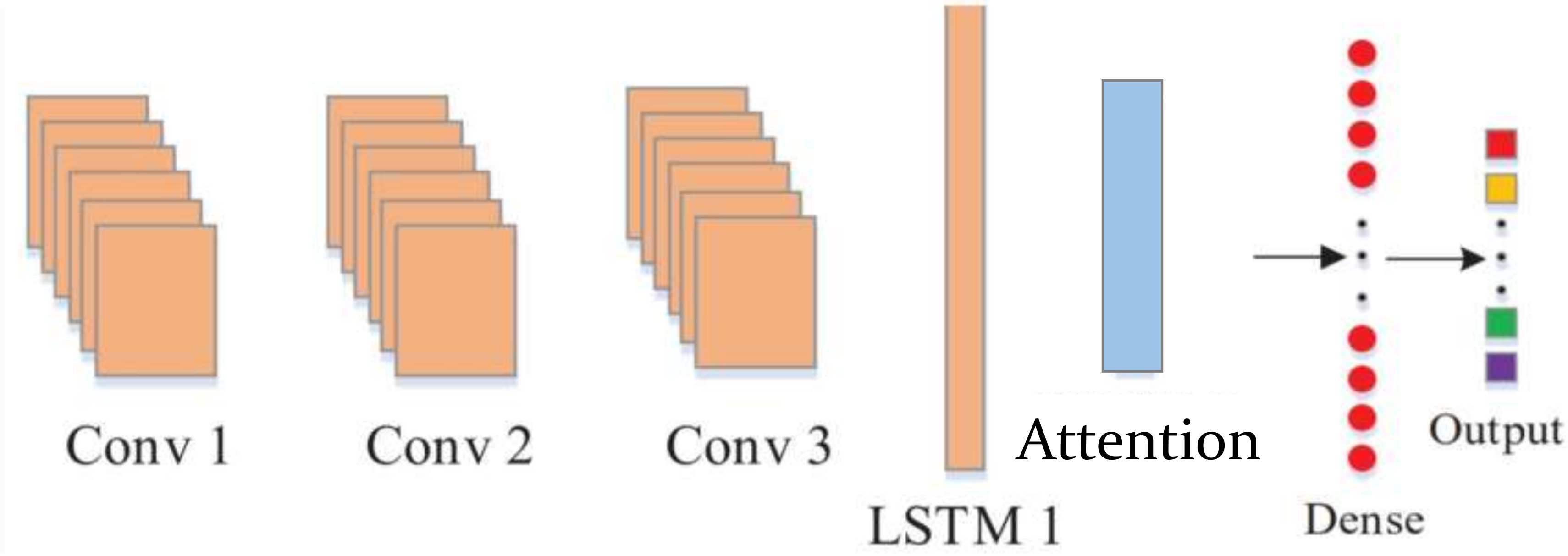
Layer (type)	Output Shape	Param #
reshape_10 (Reshape)	(None, 2, 128, 1)	0
zero_padding2d_14 (ZeroPaddi	(None, 2, 132, 1)	0
conv2d_22 (Conv2D)	(None, 2, 132, 256)	1024
max_pooling2d_31 (MaxPooling	(None, 2, 66, 256)	0
dropout_43 (Dropout)	(None, 2, 66, 256)	0
conv2d_23 (Conv2D)	(None, 2, 66, 256)	393472
max_pooling2d_32 (MaxPooling	(None, 2, 33, 256)	0
dropout_44 (Dropout)	(None, 2, 33, 256)	0
conv2d_24 (Conv2D)	(None, 2, 33, 80)	61520
max_pooling2d_33 (MaxPooling	(None, 2, 16, 80)	0
dropout_45 (Dropout)	(None, 2, 16, 80)	0
conv2d_25 (Conv2D)	(None, 2, 16, 80)	19280
max_pooling2d_34 (MaxPooling	(None, 2, 8, 80)	0
dropout_46 (Dropout)	(None, 2, 8, 80)	0
reshape_11 (Reshape)	(None, 2, 640)	0
lstm_9 (LSTM)	(None, 50)	138200
dropout_47 (Dropout)	(None, 50)	0
dense_10 (Dense)	(None, 128)	6528
dropout_48 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 11)	1419

Total params: 621,443
Trainable params: 621,443
Non-trainable params: 0

CNN-LSTM: Training Performance



CNN-LSTM (Attention): General Architecture



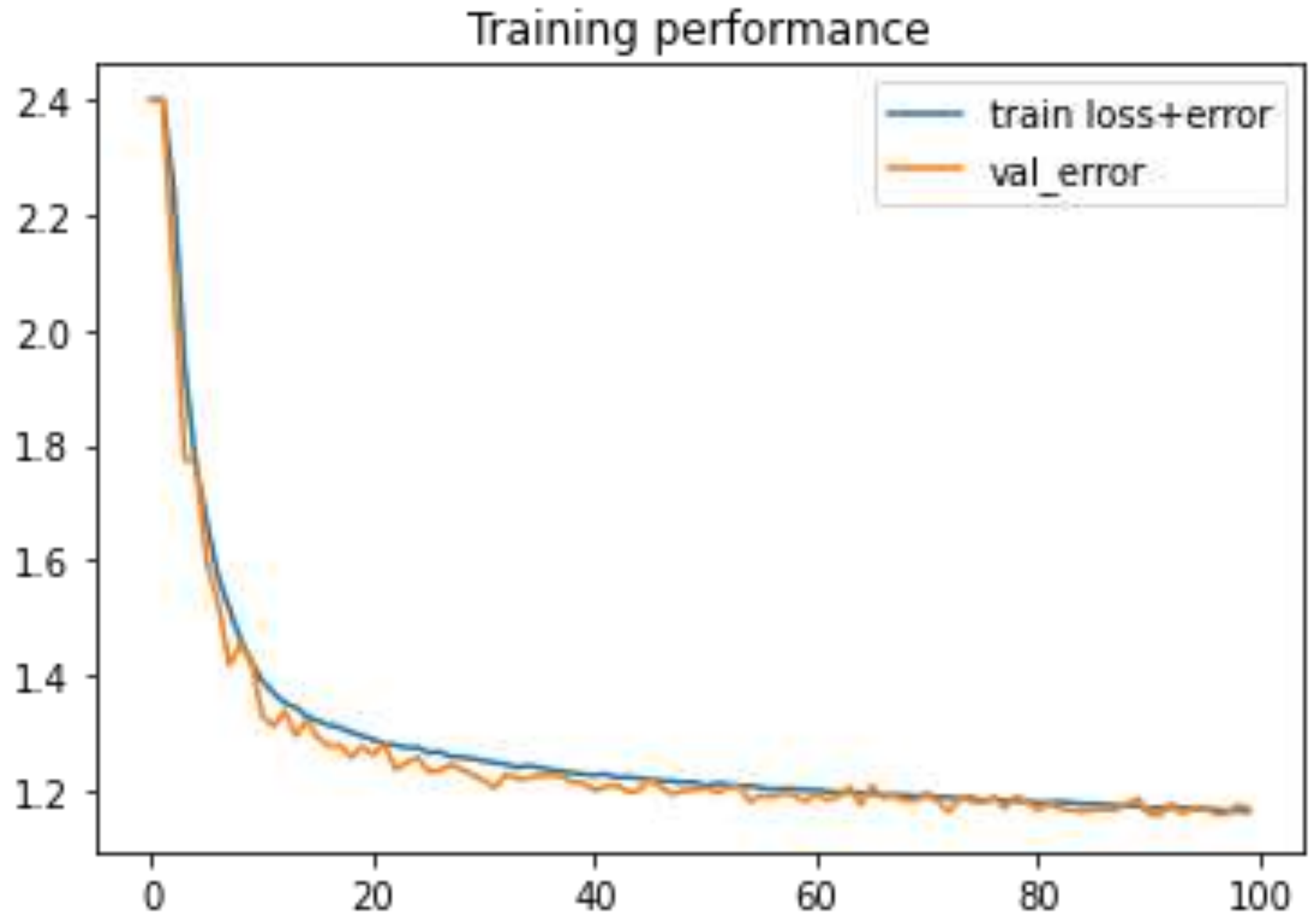
Zhang, Zufan, et al. "Automatic Modulation Classification Using CNN-LSTM Based Dual-Stream Structure." *IEEE Transactions on Vehicular Technology* 69.11 (2020): 13521-13531.

CNN-LSTM (Attention): Implemented Model

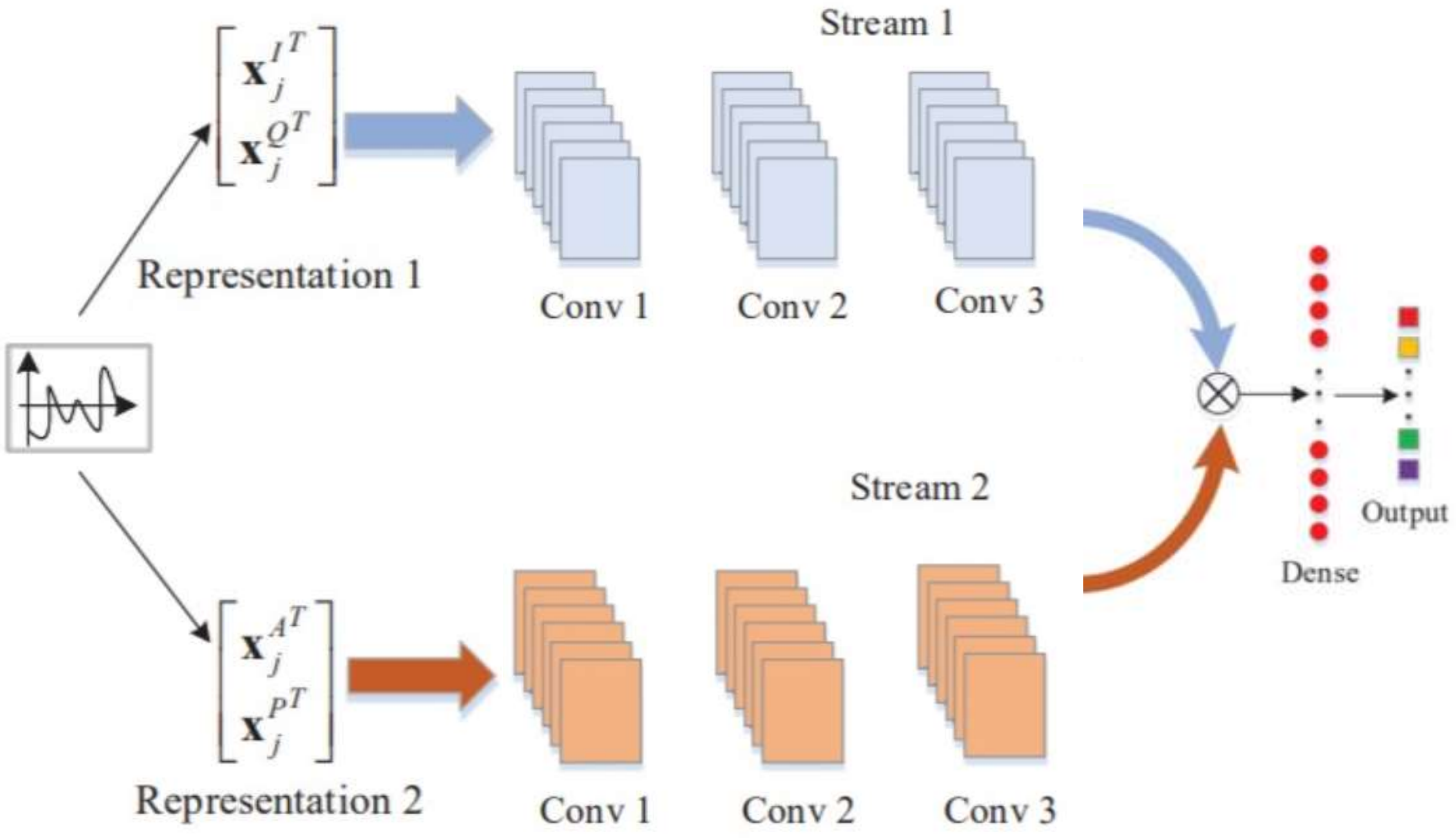
reshape_10 (Reshape)	(None, 2, 128, 1)	0
zero_padding2d_5 (ZeroPaddin	(None, 2, 132, 1)	0
conv2d_20 (Conv2D)	(None, 2, 132, 256)	1024
max_pooling2d_20 (MaxPooling	(None, 2, 66, 256)	0
dropout_27 (Dropout)	(None, 2, 66, 256)	0
conv2d_21 (Conv2D)	(None, 2, 66, 256)	393472
max_pooling2d_21 (MaxPooling	(None, 2, 33, 256)	0
dropout_28 (Dropout)	(None, 2, 33, 256)	0
conv2d_22 (Conv2D)	(None, 2, 33, 80)	61520
max_pooling2d_22 (MaxPooling	(None, 2, 16, 80)	0
dropout_29 (Dropout)	(None, 2, 16, 80)	0
conv2d_23 (Conv2D)	(None, 2, 16, 80)	19280
max_pooling2d_23 (MaxPooling	(None, 2, 8, 80)	0
dropout_30 (Dropout)	(None, 2, 8, 80)	0
reshape_11 (Reshape)	(None, 2, 640)	0
lstm_5 (LSTM)	(None, 2, 50)	138200
attention_2 (attention)	(None, 50)	52
dropout_31 (Dropout)	(None, 50)	0
dense_7 (Dense)	(None, 128)	6528
dropout_32 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 11)	1419

Total params: 621,495
Trainable params: 621,495
Non-trainable params: 0

CNN-LSTM (Attention): Training Performance



Dual Stream CNN: General Architecture



Zhang, Zufan, et al. "Automatic Modulation Classification Using CNN-LSTM Based Dual-Stream Structure." *IEEE Transactions on Vehicular Technology* 69.11 (2020): 13521-13531.

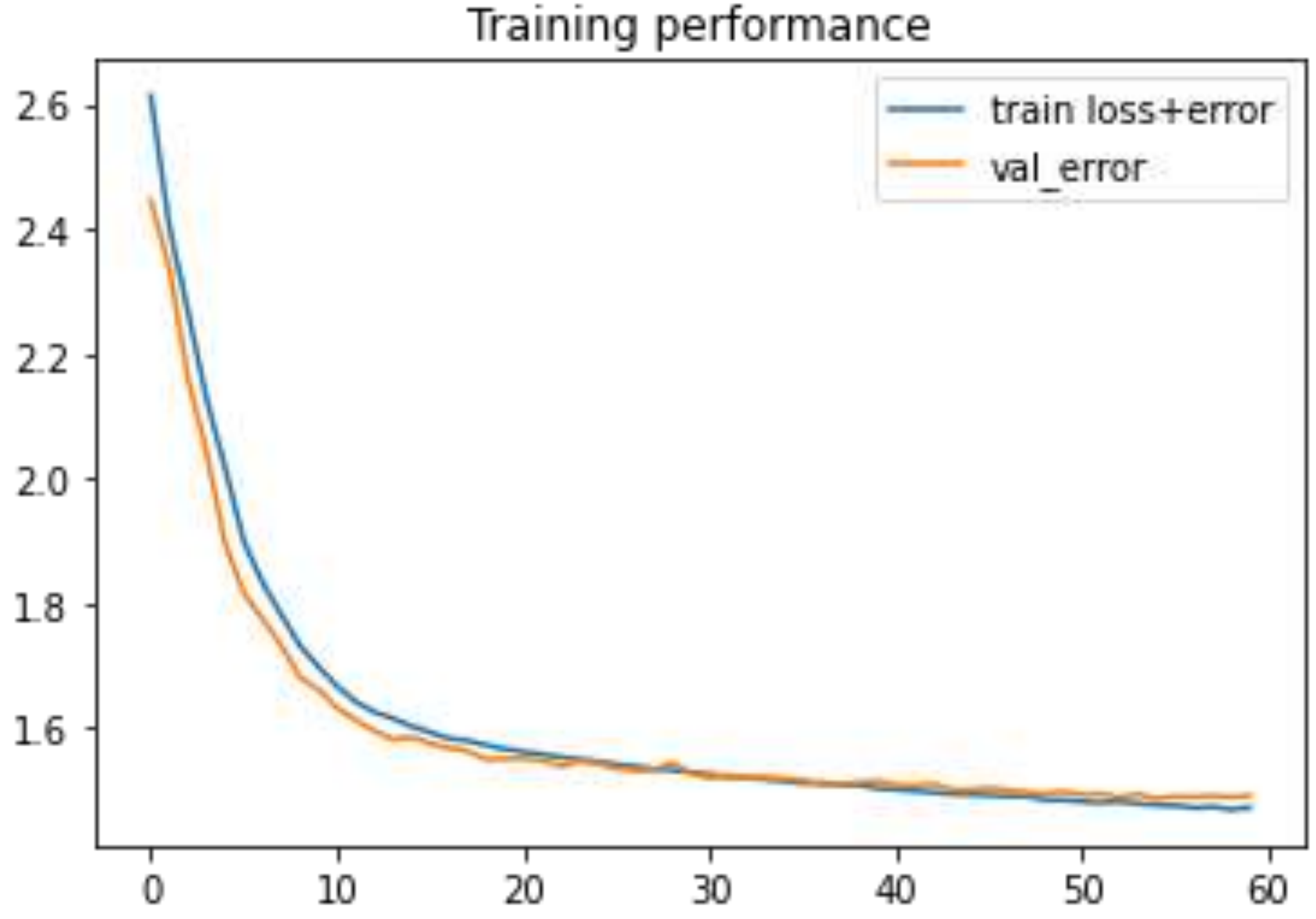
Dual Stream CNN: Implemented Model

Model: "model_2"

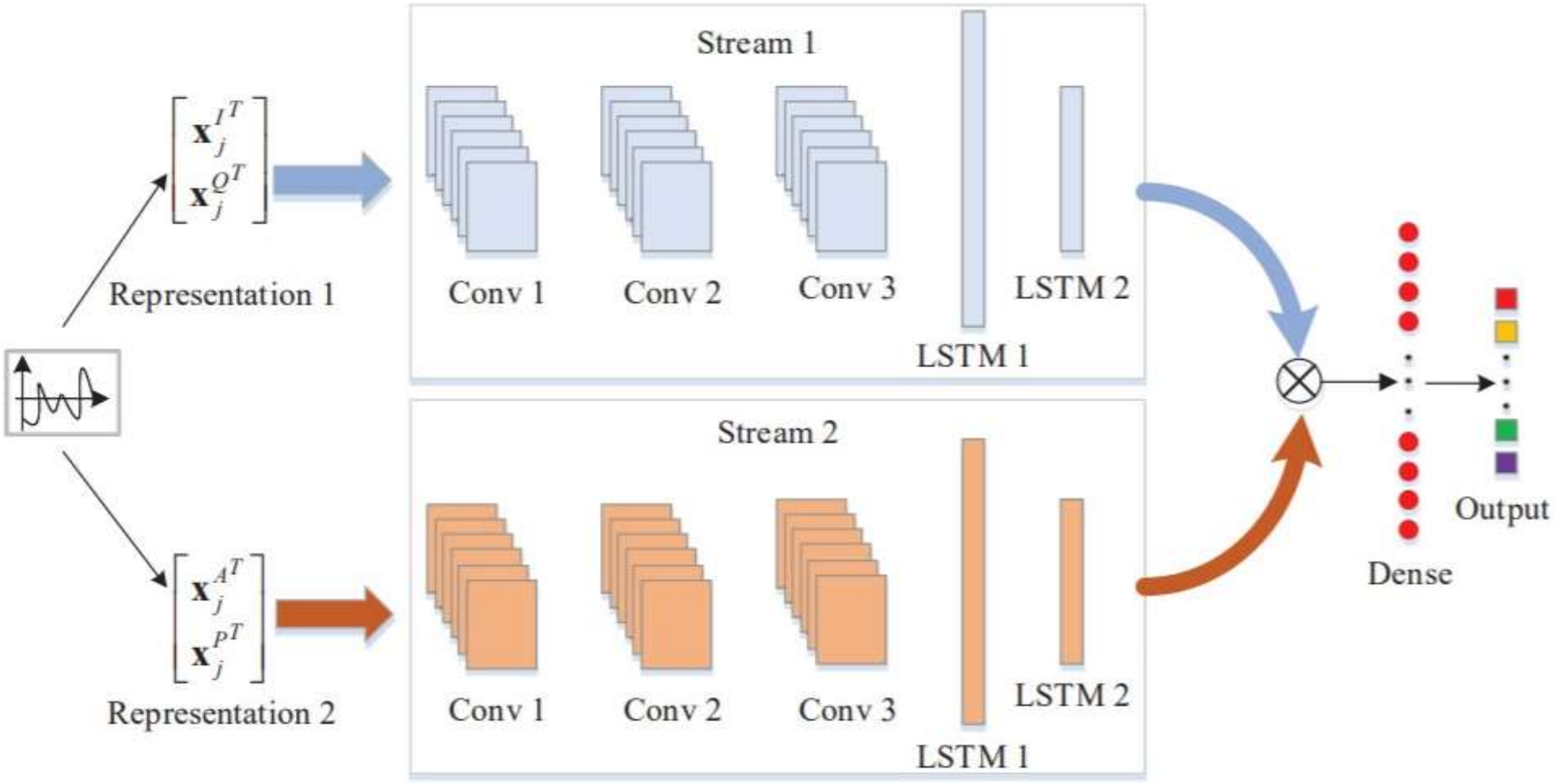
Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 2, 128, 1)]	0	
input_3 (InputLayer)	[(None, 2, 128, 1)]	0	
zero_padding2d_3 (ZeroPadding2D	(None, 2, 132, 1)	0	input_2[0][0]
zero_padding2d_6 (ZeroPadding2D	(None, 2, 132, 1)	0	input_3[0][0]
conv1 (Conv2D)	(None, 2, 130, 256)	1024	zero_padding2d_3[0][0]
conv11 (Conv2D)	(None, 2, 130, 256)	1024	zero_padding2d_6[0][0]
dropout_2 (Dropout)	(None, 2, 130, 256)	0	conv1[0][0]
dropout_5 (Dropout)	(None, 2, 130, 256)	0	conv11[0][0]
zero_padding2d_4 (ZeroPadding2D	(None, 2, 130, 260)	0	dropout_2[0][0]
zero_padding2d_7 (ZeroPadding2D	(None, 2, 130, 260)	0	dropout_5[0][0]
conv2 (Conv2D)	(None, 1, 128, 256)	399616	zero_padding2d_4[0][0]
conv22 (Conv2D)	(None, 1, 128, 256)	399616	zero_padding2d_7[0][0]
dropout_3 (Dropout)	(None, 1, 128, 256)	0	conv2[0][0]
dropout_6 (Dropout)	(None, 1, 128, 256)	0	conv22[0][0]
conv3 (Conv2D)	(None, 1, 126, 80)	61520	dropout_3[0][0]
conv33 (Conv2D)	(None, 1, 126, 80)	61520	dropout_6[0][0]
dropout_4 (Dropout)	(None, 1, 126, 80)	0	conv3[0][0]
dropout_7 (Dropout)	(None, 1, 126, 80)	0	conv33[0][0]
concatenate (Concatenate)	(None, 1, 126, 160)	0	dropout_4[0][0] dropout_7[0][0]
flatten (Flatten)	(None, 20160)	0	concatenate[0][0]
dense (Dense)	(None, 256)	5161216	flatten[0][0]
dropout_8 (Dropout)	(None, 256)	0	dense[0][0]
dense_1 (Dense)	(None, 11)	2827	dropout_8[0][0]

Total params: 6,088,363
Trainable params: 6,088,363
Non-trainable params: 0

Dual Stream CNN: Training Performance



Dual Stream CNN-LSTM: General Architecture



Zhang, Zufan, et al. "Automatic Modulation Classification Using CNN-LSTM Based Dual-Stream Structure." *IEEE Transactions on Vehicular Technology* 69.11 (2020): 13521-13531.

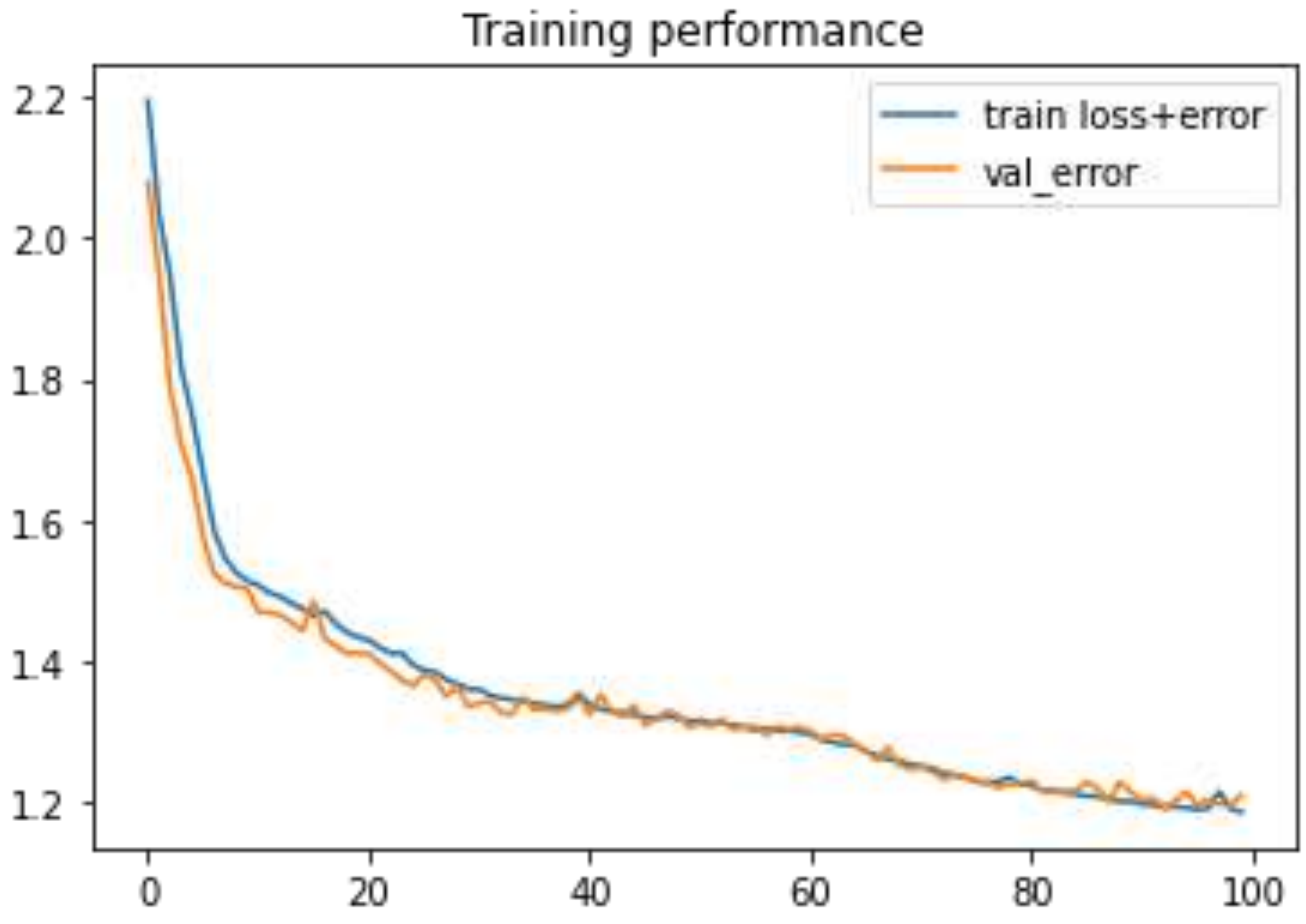
Dual Stream CNN-LSTM: Implemented Model (1)

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 2, 128, 1)]	0	
input_2 (InputLayer)	[(None, 2, 128, 1)]	0	
zero_padding2d (ZeroPadding2D)	(None, 2, 132, 1)	0	input_1[0][0]
zero_padding2d_4 (ZeroPadding2D	(None, 2, 132, 1)	0	input_2[0][0]
conv1 (Conv2D)	(None, 2, 130, 256)	1024	zero_padding2d[0][0]
conv11 (Conv2D)	(None, 2, 130, 256)	1024	zero_padding2d_4[0][0]
dropout (Dropout)	(None, 2, 130, 256)	0	conv1[0][0]
dropout_5 (Dropout)	(None, 2, 130, 256)	0	conv11[0][0]
zero_padding2d_1 (ZeroPadding2D	(None, 2, 130, 260)	0	dropout[0][0]
zero_padding2d_5 (ZeroPadding2D	(None, 2, 130, 260)	0	dropout_5[0][0]
conv2 (Conv2D)	(None, 1, 128, 256)	399616	zero_padding2d_1[0][0]
conv22 (Conv2D)	(None, 1, 128, 256)	399616	zero_padding2d_5[0][0]
dropout_1 (Dropout)	(None, 1, 128, 256)	0	conv2[0][0]
dropout_6 (Dropout)	(None, 1, 128, 256)	0	conv22[0][0]
conv3 (Conv2D)	(None, 1, 126, 80)	61520	dropout_1[0][0]
conv33 (Conv2D)	(None, 1, 126, 80)	61520	dropout_6[0][0]
dropout_2 (Dropout)	(None, 1, 126, 80)	0	conv3[0][0]
dropout_7 (Dropout)	(None, 1, 126, 80)	0	conv33[0][0]

Dual Stream CNN-LSTM: Implemented Model(2)

dropout_7 (Dropout)	(None, 1, 126, 80)	0	conv33[0][0]
zero_padding2d_3 (ZeroPadding2D	(None, 1, 134, 80)	0	dropout_2[0][0]
zero_padding2d_7 (ZeroPadding2D	(None, 1, 134, 80)	0	dropout_7[0][0]
reshape (Reshape)	(None, 134, 80)	0	zero_padding2d_3[0][0]
reshape_2 (Reshape)	(None, 134, 80)	0	zero_padding2d_7[0][0]
lstm (LSTM)	(None, 134, 100)	72400	reshape[0][0]
lstm_2 (LSTM)	(None, 134, 100)	72400	reshape_2[0][0]
dropout_3 (Dropout)	(None, 134, 100)	0	lstm[0][0]
dropout_8 (Dropout)	(None, 134, 100)	0	lstm_2[0][0]
lstm_1 (LSTM)	(None, 50)	30200	dropout_3[0][0]
lstm_3 (LSTM)	(None, 50)	30200	dropout_8[0][0]
dropout_4 (Dropout)	(None, 50)	0	lstm_1[0][0]
dropout_9 (Dropout)	(None, 50)	0	lstm_3[0][0]
reshape_1 (Reshape)	(None, 1, 50)	0	dropout_4[0][0]
reshape_3 (Reshape)	(None, 50, 1)	0	dropout_9[0][0]
multiply (Multiply)	(None, 50, 50)	0	reshape_1[0][0] reshape_3[0][0]
flatten (Flatten)	(None, 2500)	0	multiply[0][0]
dense (Dense)	(None, 11)	27511	flatten[0][0]

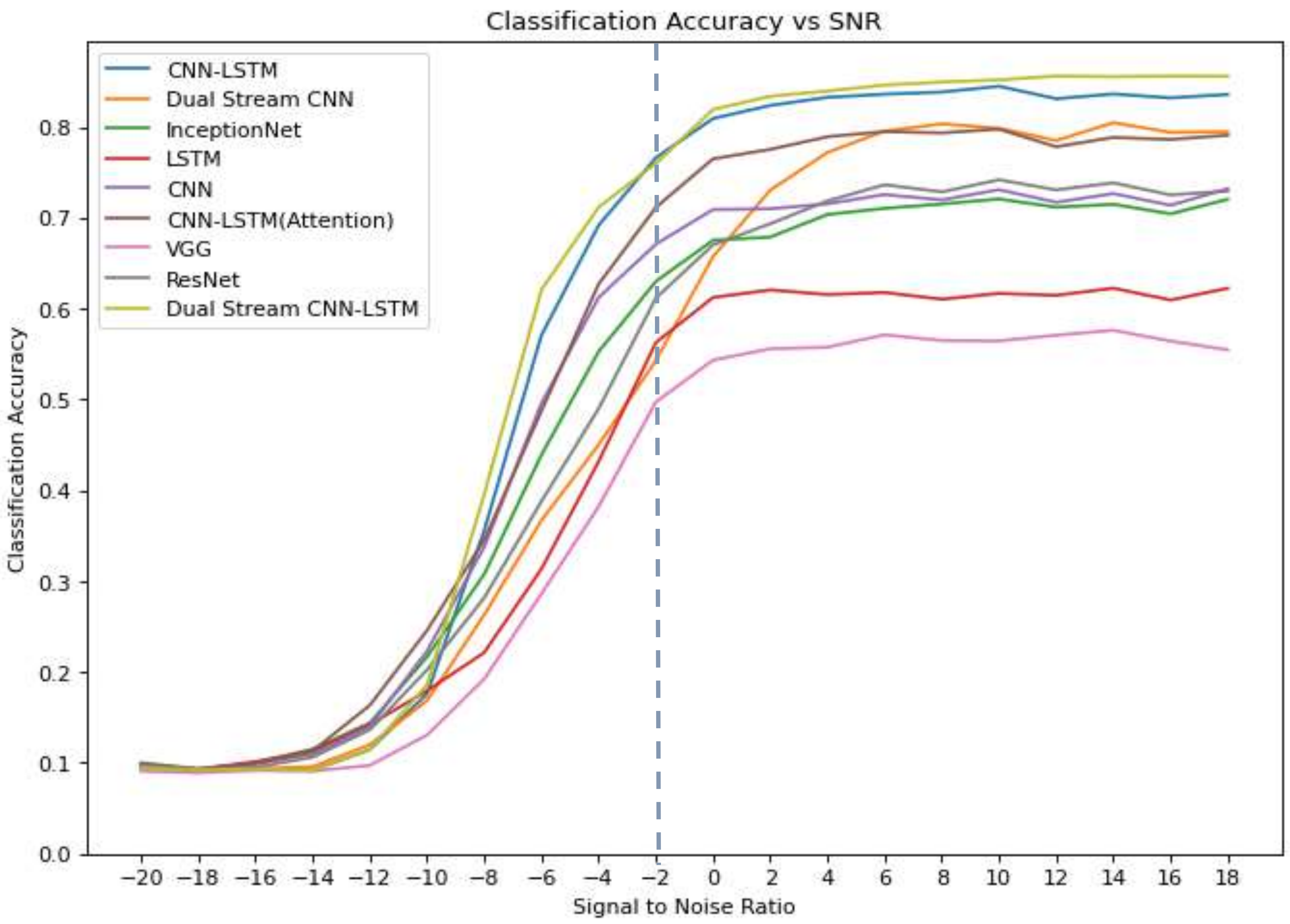
Dual Stream CNN-LSTM: Training Performance



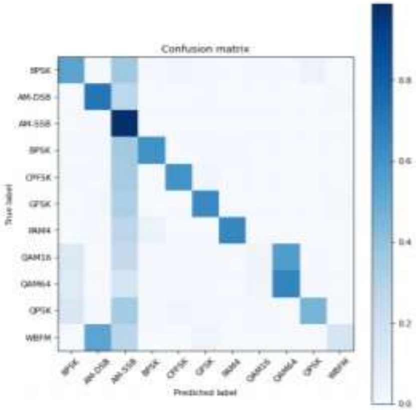
Results: Average Accuracy

Model Name	Average Accuracy
CNN	50.39054382895153
LSTM	42.09282126493383
CNN-LSTM	56.83105409081219
ResNet	48.57032160678825
VGG	37.8584162464500
InceptionNet	48.76701468450949
Dual Stream CNN	50.1184001948336
Dual Stream CNN-LSTM	58.11448660169753
CNN-LSTM with Attention	54.2070577385623

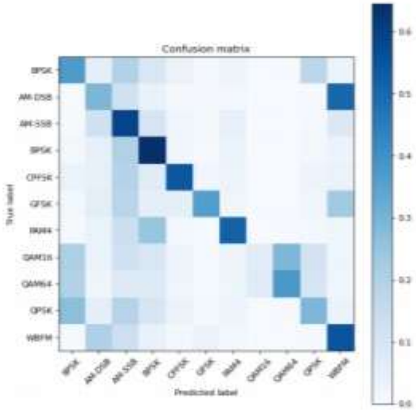
Results: SNR vs Accuracy



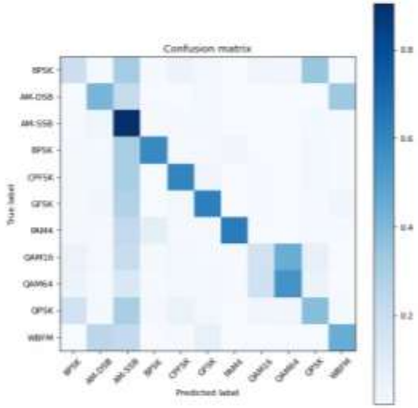
Results: Confusion Matrices



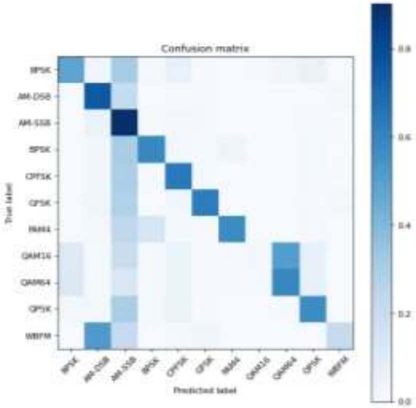
(a) CNN-LSTM(Att)



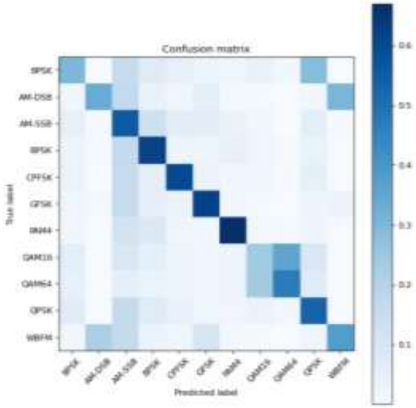
(b) LSTM



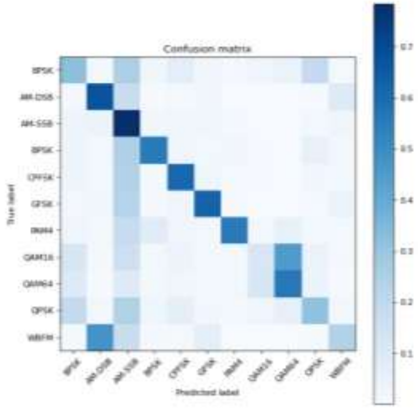
(c) CNN



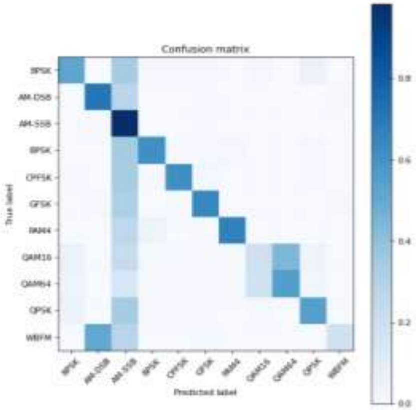
(d) DS CNN-LSTM



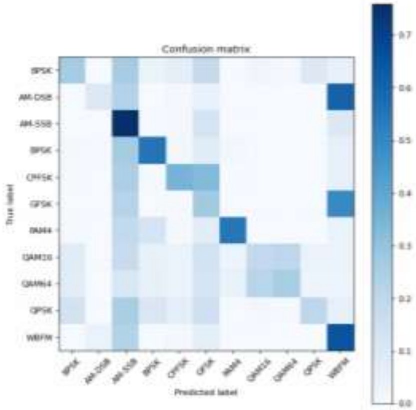
(e) ResNet



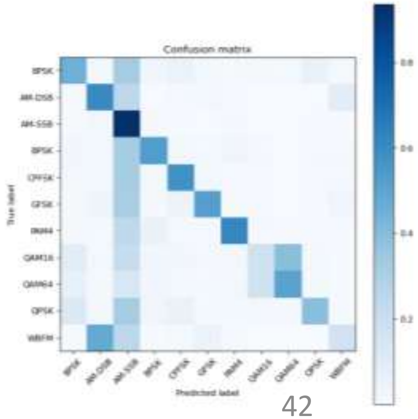
(f) InceptionNet



(g) CNN-LSTM



(h) VGG



(i) DS CNN

Conclusion

- Breadth first implementation of 9 architectures, in order to find model most suitable for development.
- Classification accuracy of our models is lower than published models with similar architectures.
- Validation accuracy is not a simple function of the depth of the Neural Network but of the features learned by these layers.
- Dual Stream CNN-LSTM has the best validation accuracy
- CNN-LSTM based architectures perform much better than other architectures.
  - Due to temporal & spatial variations of radio signals.

Conclusion

- Confusion matrices show :
  - that AM – SSB is mostly misclassified
  - QAM16 is often not classified.
- Sharp Drop off in Accuracy from 0 SNR begins.
  - Highlights need for denoising

- Dual Stream architectures allow us to train network on both In-phase & Quadrature Data as well as Amplitude & Phase Data
  - Combining Dual Stream architecture with CNN-LSTM networks gives network more information
  - Cleaner diagonals for CNN-LSTM based networks

Future Work

- Exhaustive exploration of hyperparameters

- Run on larger datasets made available by DeepSig.
  - More modulation types
  - More data per modulation type

- Need for denoising and signal correction
  - Use of Correctional Neural Networks
    - Phase offset
    - Frequency Offset

  - Denoising Auto-Encoders
    - IQ data
    - Separate IQ & AP Auto- Encoders for Dual stream networks
    - Stacked Autoencoders

Future Work

- Explore use of only Amplitude Phase data
- Explore BiLSTM, and Deep Complex Networks
- Further exploration of Attention modules
- Need for networks that can resample, synchronize, and remove non-linear channel distortions for real world use
- Explore implementation on edge and/or with Software Defined Radio (GNURadio)