Enhanced Low SNR Radio Signal Classification using Transformer based Deep Learning

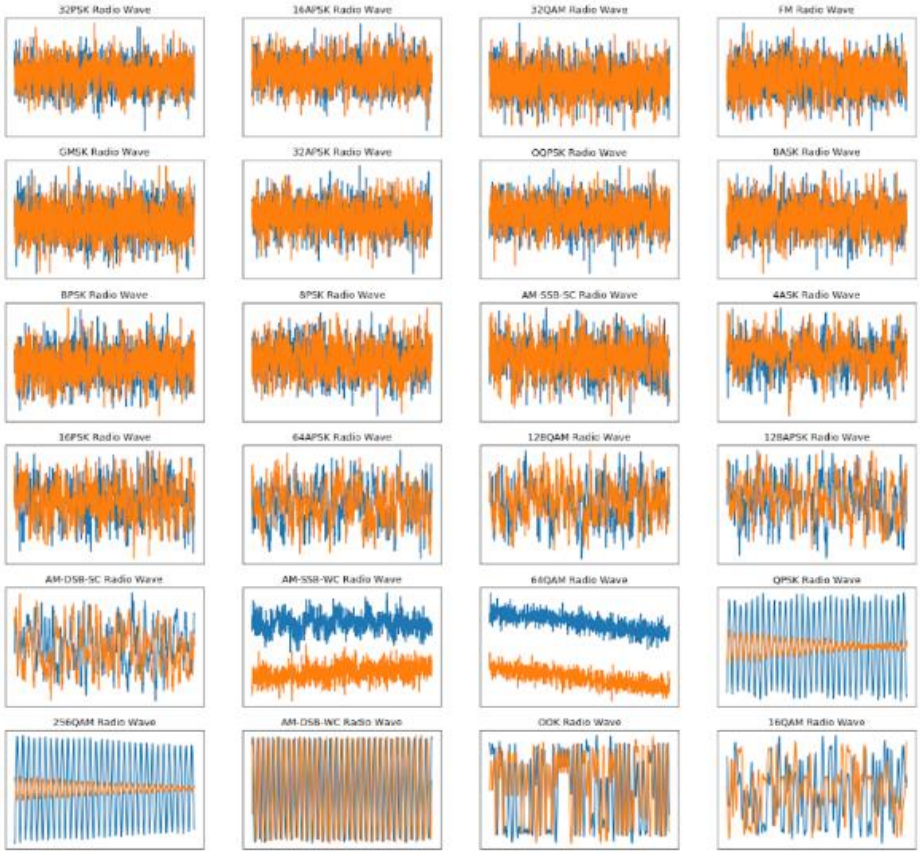
Abstract

The ability to classify signals is an important task that holds the opportunity for many different applications. Previously to classify the signal, we should decompose the signal using FT (Fourier Transform), SIFT, MFCC, or another handcrafting method using statistical modulation features. In the past five years, we have seen rapid disruption occurring based on the improved neural network architectures, algorithms, and optimization techniques collectively known as deep learning (DL). It turns out that state of the art deep learning methods can be applied to the same problem of signal classification and shows excellent results while completely avoiding the need for difficult handcrafted feature selection. In 2018, people use ResNet as a state of the art of computer vision to classify radio communication signals. But ResNet only still fails to distinguish signals with low SNR conditions. They only work well on a signal with high SNR Conditions. After two years, deep learning has already improved a lot and many methods have become the new state of the art that we could apply for radio signal classification. Hence, we propose a new state of the art method to better classify radio-signal networks that both work on a signal with low noise (High SNR) and signal with high noise (Low SNR). Our works even work using only RAW signals without the need of preprocessing or denoising the noisy signal.

I. Background

Rapidly understanding and labeling the radio spectrum in an autonomous way is a key enabler for spectrum interference monitoring, radio fault detection, dynamic spectrum access, opportunistic mesh networking, and numerous regulatory and defense applications. Today's military operations depend on the extensive use of wireless communication technologies. Monitoring of radio signals may reveal vital information regarding the detection, localization, and identification of an opponent. Traditionally, operators who were trained to recognize various signal formats based on manual ‘listen in’ techniques performed the identification.

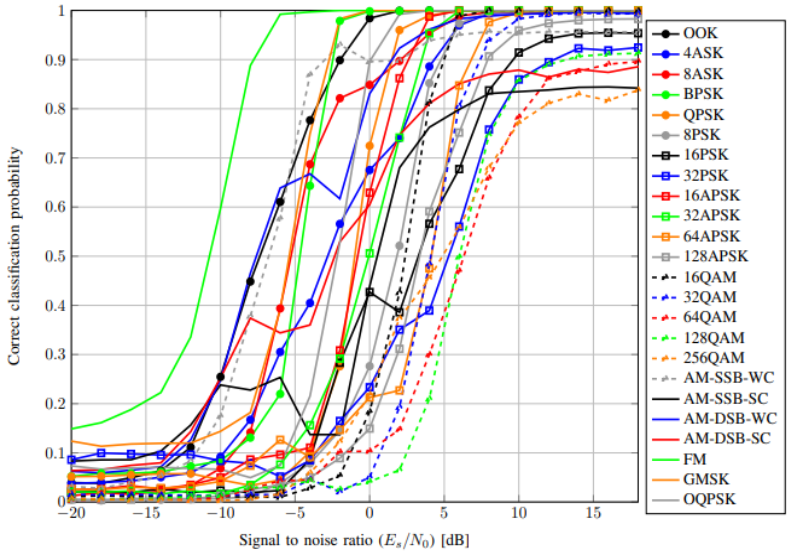
For many years, radio signal classification and modulation recognition have been accomplished by carefully handcrafting specialized feature extractors for specific signal types and properties and by deriving compact decision bounds from them using either analytically derived decision boundaries or statistical learned boundaries within low-dimensional feature spaces. Consider the image below: these are just a few of the many possible signals that a machine may need to differentiate.



For humans, it is really difficult to differentiate the signal by a look at each signal with our eyes. Hence to classify those signals, humans need to extract some features first. The first method is using statistical modulation features. Using this method, we need to extract the structure of the carrier, symbol timing, and symbol structure for certain modulations and then move to the next step of decision criterion using the machine learning method. These methods work well and successfully provide a robust classification for the signal itself. But it still needs a lot of information to classify the signal, hence it will be difficult if the information we got is not completed or one of the information is missing. The second method is Radio Channel Models, although it is easier than the first method to make a stochastic model, we still need to create the model of the signal first. We could not input the raw signal directly to our system and get the classification result.

In the past few years, **deep learning models have out-paced traditional methods in computer vision** such that, like the current state of signal classification, involved meticulously creating hand-crafted feature extractors. Deep learning provides a hands-off approach that allows us to automatically learn important features directly from the raw data. In 2018, people used the deep learning method (ResNet) to classify radio signals. They got a nice result with an accuracy of almost 94% for high SNR signals. This method works even with RAW signal without

hand-crafting, denoising or pre-processing the signal. This then proved that Deep Learning could become new state-of-the-art in radio signal classification. The method even does not need pre-processing and de-noising every signal. But there is still some weakness, that the method is not doing good with low SNR signal (Signal with really high noise). The performance could be shown below:



We could see from the figure that the signal classification still has the problem of classifying the low SNR signal. The accuracy for certain signal under 0 dB decreases a lot, even the accuracy is under 50%. **In this proposal, we would create a new state-of-the-art method that could classify low SNR and high SNR signals with better accuracy**. The result will help a lot for classifying signals in real condition since we will not always receive an ideal signal in real life.

II. Problem Statement

1. **What is the new state of the art of deep learning needs to be chosen and suitable for radio signal classification?**

   The latest Radio Signal Classification using Deep Learning is done in 2018. After two years there is a lot of new state-of-the-art Computer Vision including new networks, new optimization, and even a new type of regularization. We could use this new state of the art to produce more robust and more efficient communication signal classification.

2. **How could we make use of a large dataset to classify the RAW signal using the new state of the art of deep learning network?**

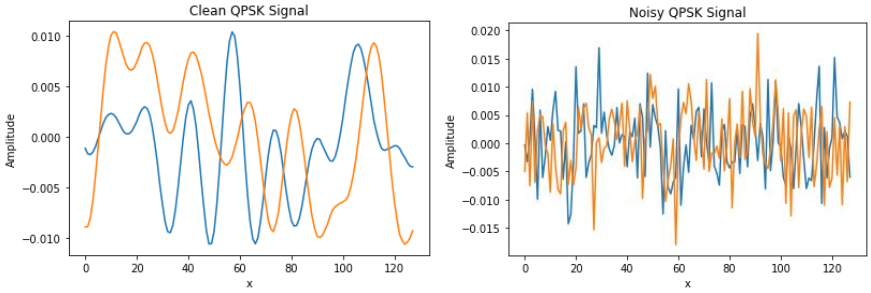
   We use a dataset from DeepSig that contains the representation of many different kinds of communication signals. This dataset contains both clean signals and noisy signals. In real life, the signal is always not in the ideal state, we could not mimic a clean signal directly for this classification, because the model created from a clean signal only will be difficult to recognize the signal in daily life. We need to process the

datasets and make a selection from the dataset also make use of the noisy signal to get the representation of signal both in low SNR and high SNR conditions.

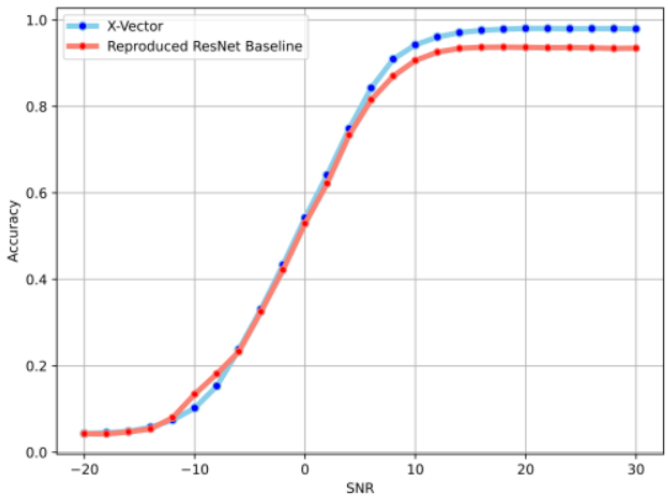
3. **How could we get higher accuracy to classify signals both in low SNR and high SNR?**
   Some of the previous work is still not good to classify signals in Low SNR, it means that if the noise is higher, the model will likely fail to do the classification. With the new state of the art of computer vision, we would improve the works, so it will be possible to classify signals with high noise without compromising the accuracy for the signal with low noise.

III. Challenges

1. In reality, the signal is always not ideal and combined with the unwanted signal that is considered as noise. In the image above, we can see how drastically noise can affect our ability to recognize a signal.



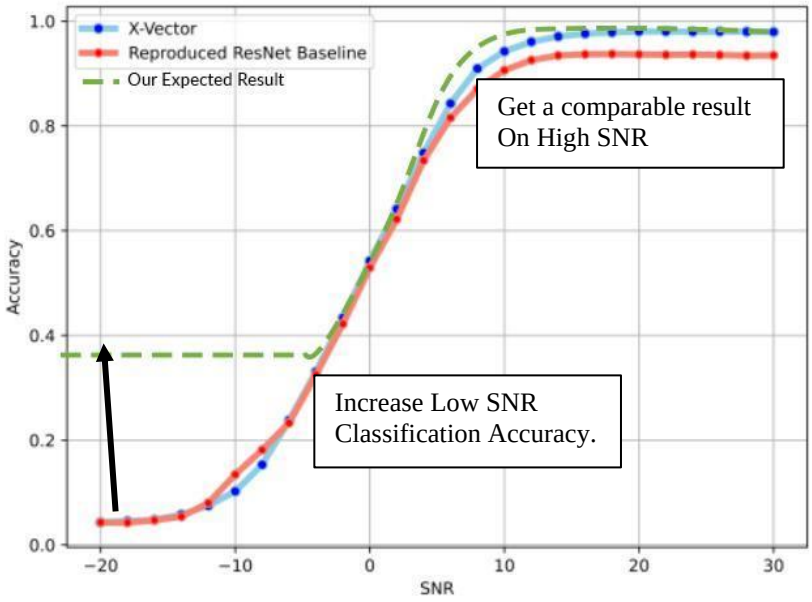
2. A clean signal will have a high SNR and a noisy signal will have a low SNR. We should be able to classify signals both in low SNR and high SNR using the RAW signal without preprocessing needed. Denoising or preprocessing many kinds of signal is not an easy task, because some signal needs different treatment in case of preprocessing, sometimes we could not do the preprocessing to the signal that we do not know what the signal is. Hence, one of the challenges is how we could classify the RAW signal directly.



3. We use the DeepSig Radio Signal Dataset, this dataset is pretty large (18GB) with consists 24 types of

signal modulation such as 32PSK,16APSK, 32QAM,FM,GMSK, 32APSK, OQPSK, 8ASK, BPSK, 8PSK, AM-SSB-SC, 4ASK, 16PSK, 64APSK, 128QAM,128APSK, AM-DSB-SC, AM-SSB-WC, 64QAM, QPSK, 256QAM, AM-DSB-WC, OOK, and 16QAM . We need to determine what is the best network that could classify both low and high SNR signals faster.

IV. Goal



Our Expected Result compare to the SOTA

Previous research using ResNet already got good accuracy in the case of classifying ideal signal with less noise. But ResNet still has low performance in case of classifying signals with high noise. In a real application, we could not always hope that we will receive an ideal signal with low noise, we also often receive a signal with high noise. That is why our goals will be focused on classifying signals both in low and high SNR. Here is the summary of our goals:

- Our target is to **get better accuracy in lower SNR signals without sacrifice accuracy in higher SNR signals**. It will indicate that our model is robust under the noisy signal modulation. Then our method could be used in real-condition where we do not always receive an ideal clear signal without noise.

- We will design a new deep learning architecture and try **to get comparable results in terms of accuracy with state of the art or even better.**

- Our deep learning method **will use the RAW signal directly as an input**, since preprocessing and denoise the signal needs different treatment according to what kind of signal is. Hence, in some cases, we could not always use the preprocessed signal as our input.

V. Method

i. Previous Method

Baseline ResNet Architecture [1]

Layer	Output Dimensions
Input	2 x 1024
Residual Stack	32 x 512
Residual Stack	32 x 256
Residual Stack	32 x 128
Residual Stack	32 x 64
Residual Stack	32 x 32
Residual Stack	32 x 16
FC/SeLU	128
FC/SeLU	128
FC/Softmax	24

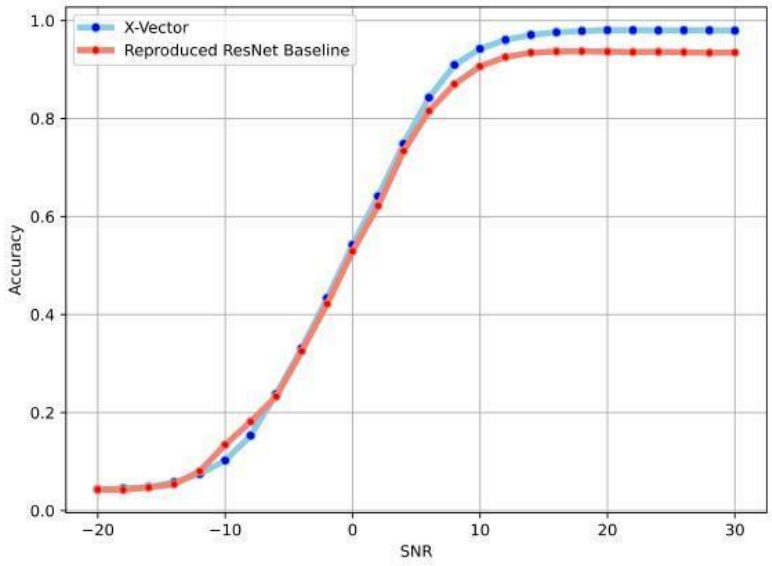
Maximum Accuracy : 93.7 %

X-Vector Architecture [2]

Layer	Output Dimensions
Input	2 x 1024
Conv 1D (ReLU)	64 x 1024
Conv 1D (ReLU)	64 x 1024
Conv 1D (ReLU)	64 x 1024
Conv 1D (ReLU)	64 x 1024
Conv 1D (ReLU)	64 x 1024
Conv 1D (ReLU)	64 x 1024
Conv 1D	64 x 1024
Average Pooling 1D	64
Variance Pooling 1D	64
Concatenate	128
FC/SeLU	128
FC/SeLU	128
FC/Softmax	24

Maximum Accuracy : 98%

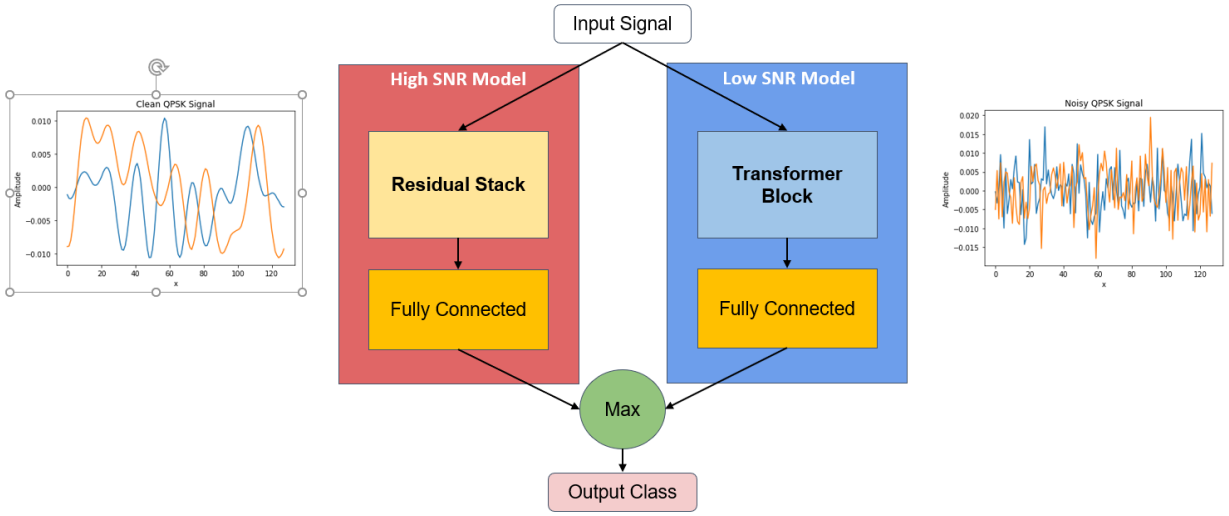
Previous method [1] uses baseline ResNet architecture and got maximum accuracy of 93.7%. New paper from the ASILOMAR Conference on Signals, Systems, and Computers in 2020 [2] got better accuracy at 98%. But we would use the first reference as our baseline since in the second reference they did not provide more detailed00000000000000000000 information about how they trained neural networks; hence we could not use their experiment as our baseline.



Baseline result compared to X-Vector result

We could see in the figure above that the result is only good for signals with high SNR, but for the signal with low SNR seems not really good. From this weakness, we then proposed our method that has better accuracy in lower SNR signal but still could get a comparable result at high SNR signal.

ii. Proposed Method



Proposed method using two neural network and ensemble network

In our proposed method, we try to use two neural network models, one is good for predicting signal with high SNR and the other one is good for predicting signals with low SNR. For high SNR models, we will use a Residual Net like our baseline method and do some modification and tuning to increase its accuracy. For low SNR models, we will use a transformer neural network, this is one kind of time series neural network that could help to predict complex data for example signals with low SNR rate. After input our signal to two models, we will ensemble by seek the maximum accuracy output resulting from the two models.

a) High SNR Model (Modified ResNet)

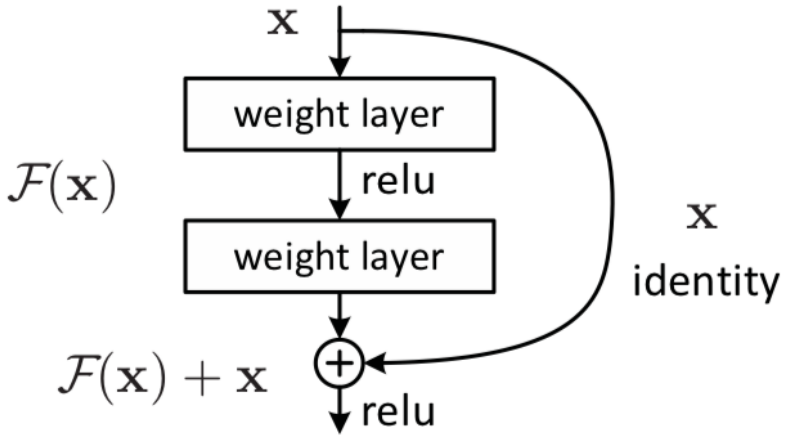
Residual Net Explained

Before explaining our model, we will explain what are the advantages of using residual nets. In recent years, neural networks have become deeper, with state-of-the-art networks going from just a few layers (e.g., AlexNet) to over a hundred layers. One of the major benefits of a very deep network is that it can represent very complex functions. However, a huge barrier to training them is vanishing gradients: very deep networks often have a gradient signal that goes to zero quickly, thus making gradient descent prohibitively slow. More specifically, during gradient descent, as we backprop from the final layer back to the first layer, we are multiplying by the weight matrix on each step. If the gradients are small, due to large number of multiplications, the gradient can decrease exponentially quickly to zero (or, in rare cases, grow exponentially quickly and “explode” to take very large values).

The regular networks like VGG-16 are called “plain” networks. In plain networks, as the number of layers increases from 20 to 56 (as shown below), even after thousands of iterations, training error was worse for a 56-layer compared to a 20-layer network. In plain networks, as the number of layers increases from 20 to 56 (as shown below), even after thousands of iterations, training error was worse for a 56-layer compared to a 20-layer network.

When deeper networks are able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades

rapidly. Using deeper networks is degrading the performance of the model. Microsoft Research paper tries to solve this problem using a Deep Residual learning framework.



Skip Connection in Residual Network

The approach is to add a shortcut or a skip connection that allows information to flow, well just say, more easily from one layer to the next layer, i.e., you bypass data along with normal CNN flow from one layer to the next layer after the immediate next. Two take away from the residual block, first Adding additional/new layers would not hurt the model's performance as regularization will skip over them if those layers were not useful. Second, If the additional/new layers were useful, even with the presence of regularization, the weights or kernels of the layers will be non-zero and model performance could increase slightly. Therefore, by adding new layers, because of the "Skip connection" / "residual connection", it is guaranteed that the performance of the model does not decrease but it could increase slightly. By stacking these ResNet blocks on top of each other, you can form a very deep network. Having ResNet blocks with the shortcut also makes it very easy for one of the blocks to learn an identity function. This means that you can stack on additional ResNet blocks with little risk of harming training set performance.'

Using a very deep neural network in our works is necessary because we would classify 24 kinds of signals that represent a complex function. Using a short or small neural network will not solve the problem, this is why we need to solve our problem with a really deep neural network.

Optimizer : SGD (**Baseline**)

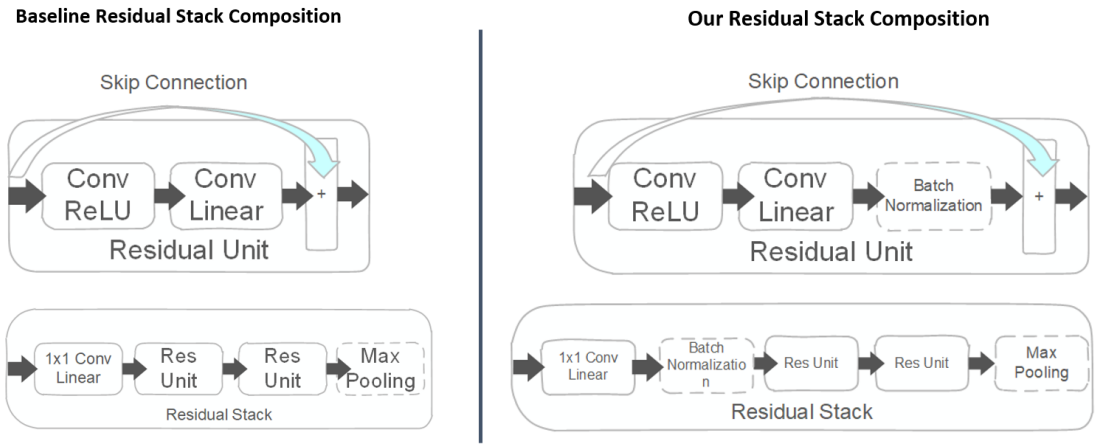
Layer	Output Dimension
Input	2 x 1024
Residual Stack	32 x 512
Residual Stack	32 x 256
Residual Stack	32 x 128
Residual Stack	32 x 64
Residual Stack	32 x 32
Residual Stack	32 x 16
FC/SeLU - Alpha Dropout	128
FC/SeLU - Alpha Dropout	128
FC/Softmax	24

Optimizer : LazyAdam (**Ours**)

Layer	Output Dimension
Input	2 x 1024
Residual Stack	32 x 512
Residual Stack	32 x 256
Residual Stack	64 x 128
Residual Stack	64 x 64
~~Residual Stack~~	~~32 x 32~~
~~Residual Stack~~	~~32 x 16~~
FC/SeLU - Dropout	128
FC/SeLU - Dropout	128
FC/Softmax	24

ResNet baseline vs Our ResNet

We do some modifications from the baseline to make the training become more efficient and improve its accuracy. We change the optimizer from **SGD** to the new state of the art of optimizer called **LazyAdam. LazyAdam** is a variant of the Adam optimizer that handles sparse updates more efficiently. The original Adam algorithm maintains two moving-average accumulators for each trainable variable, the accumulators are updated at every step. This class provides lazier handling of gradient updates for sparse variables. It only updates moving-average accumulators for sparse variable indices that appear in the current batch, rather than updating the accumulators for all indices. Compared with the original Adam optimizer, it can provide large improvements in model training throughput for some applications. We also increase the number of filters in the third and fourth residual stack both from 32 to 64. We then also remove two last residual stacks to improve the training split, we also remove those parts to avoid losing too much information because of skipping connection inside the residual network.

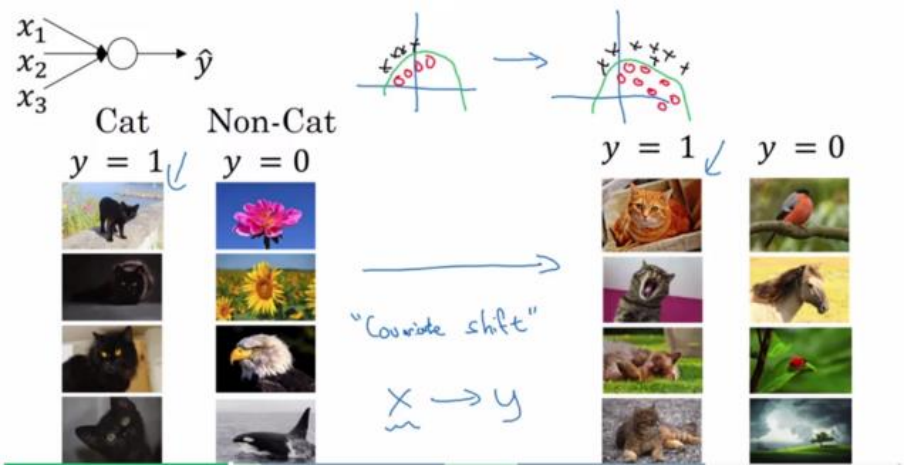


Residual Stack Baseline vs Our Residual Stack Composition

From the figure above, not only modify the neural network composition, but we also modify residual stack composition. We added batch normalization inside the residual unit, where the previous baseline not using batch normalization for their residual net. We also add batch normalization after 1x1

Conv Linear inside the residual stack.

Batch Normalization Net Explained



Batch Normalization reduces Covariance Shift

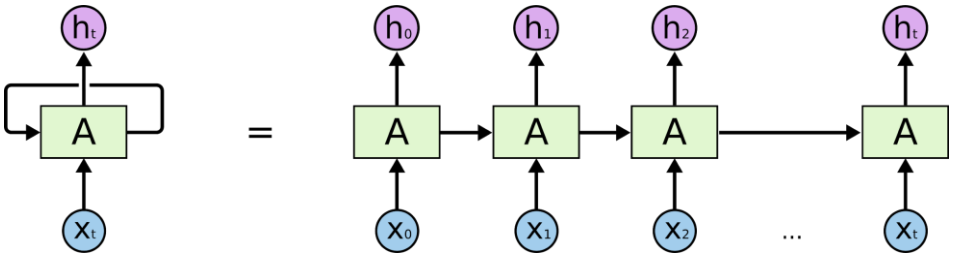
Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks. Batch normalization reduces the amount by which the hidden unit values shift around (covariance shift), for example, we train our data on only black cats' images. So, if we now try to apply this network to data with colored cats, it is obvious; we're not going to do well. Batch normalization will help this kind of problem.

Two takeaways from batch normalization, we can use higher learning rates because batch normalization makes sure **that no activation's gone too high or too low.** It reduces overfitting because it has a slight regularization effect. **Similar to residual stack, it adds some noise to each hidden layer's activations. Therefore, if we use batch normalization, we will use less residual stack, which is a good thing because we are not going to lose a lot of information.** This is the answer to why we could remove the last two residual stacks.

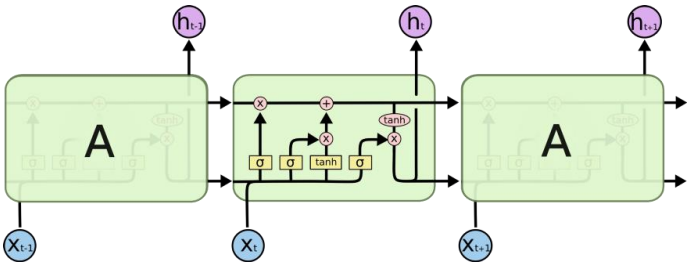
b) Low SNR Model (Transformer)

The Transformer model is the evolution of the encoder-decoder architecture, proposed in the paper titled “Attention is All You Need” [5]. While encoder-decoder architecture has been relying on recurrent neural networks (RNNs) to extract sequential information, the Transformer doesn’t use RNN. Transformer based models have primarily replaced LSTM, and it has been proved to be superior in quality for many sequence-to-sequence problems.

We do some research to find the best time-series model that was published recently. The first one we have known is the conventional method which uses recurrent neural networks (RNN), RNNs work like a feed-forward neural network that unrolls the input over its sequence, one after another.

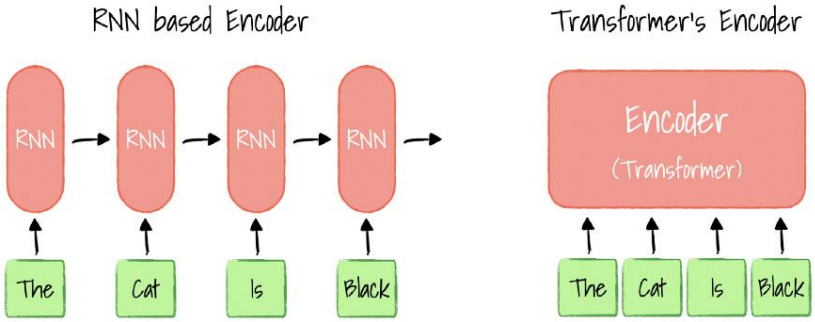


This process of unrolling each symbol in the input is done by the Encoder, whose objective is to extract data from the sequential input and encode it into a vector, a representation of the input. However, RNN models have some problems, they are slow to train because it uses sequential processing instead of parallelization, and it can’t deal with long sequences due to the vanishing gradient problem. The second one is the most used time-series model nowadays, which is Long-Short Term Memory(LSTM)networks, which are explicitly designed to avoid long-term dependency problems. Each LSTM cell allows past information to skip all the processing of the current cell and move to the next cell; this allows the memory to be retained longer and will enable data to flow along with it unchanged. LSTM consists of an input gate that decides what new information to be stored and a forget gate that determines what information to remove.

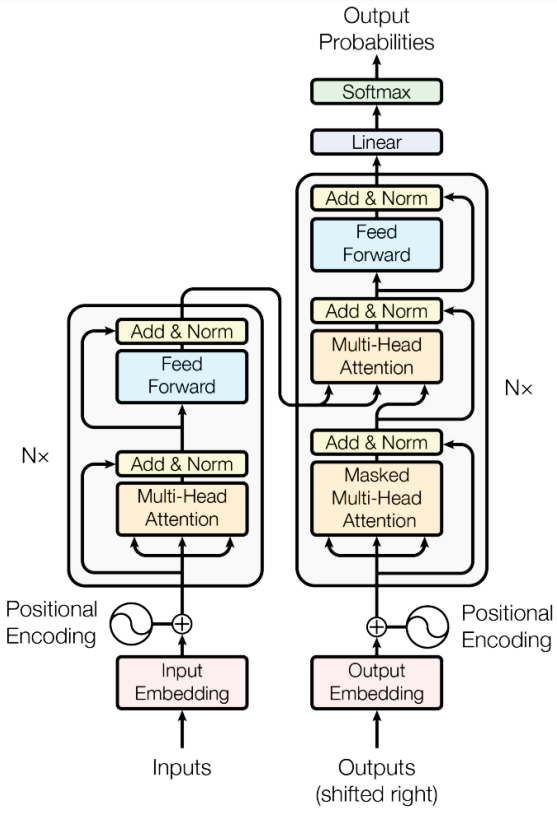


Certainly, LSTMs have improved memory, able to deal with longer sequences than RNNs. However, LSTM networks are even slower as they are more complex. Another drawback of an RNN based encoder-decoder architecture is the fixed-length vector. Using a fixed-length vector to represent the input sequence to decode an entirely new sentence is difficult. The context vector cannot store all the information if the input sequence is large. Furthermore, it is challenging to differentiate sentences with similar words but with different meanings.

And then we come up with a solution using the newest state-of-the-art time series model called Transformer. It has some advantages in using this model than previous models such as it uses parallelization processing thus make use of modern graphics processing units (GPUs), which were designed for parallel computation. It can solve the vanishing gradient problem because the input size can be any size/vector length and simultaneously proceed by the network rather than sequentially. The context of data is well captured because it uses the positional encoding and self-attention mechanism which contain in the network modules.



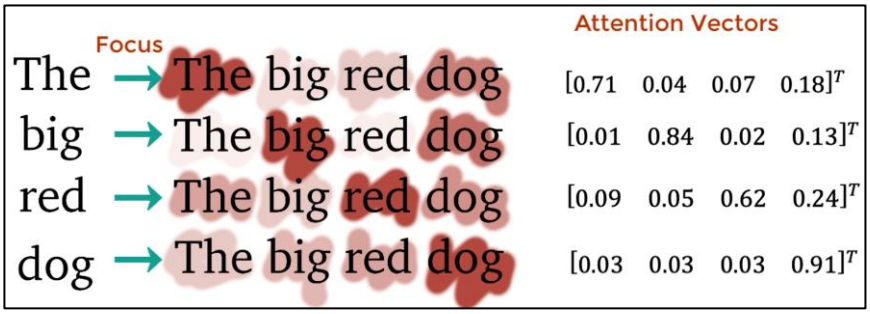
We will explain the function of the key important network's module inside the Transformer network, more detailed explanation can refer to this [5] paper. Let's assume we are training a model that translates the English sentence to French as an easier explanation for us to understand the definition. The Transformer architecture has two parts, the Encoder (left) and the Decoder (right).



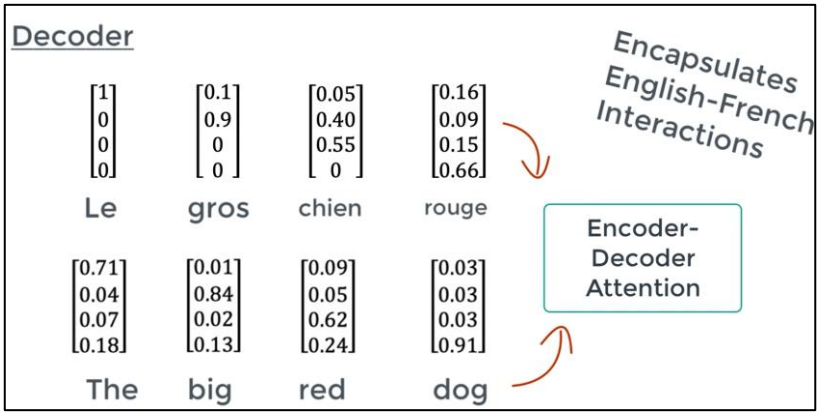
Key Transformer Modules

1. **Positional Encoding**: this module gives context based on the position of a word in a sentence.

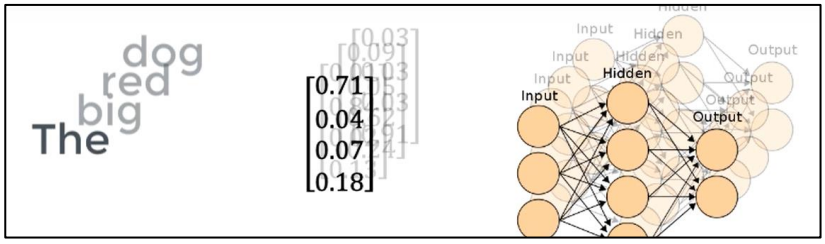
2. **Multi-Head Attention (Encoder)**: this module puts the attention mechanism on the data which determines which part of the input should the network focus on and how relevant is the i-th word/data points in the sentence/data sequence relevant to other words in the same data sequence.



3. **Multi-Head Attention (Decoder)**:this attention module will determine how related each word vector is with respect to each other, and this is where the mapping from English to French word happens.



4. **Feed-Forward net**: to transform the attention vectors into a form that is digestible by the next encoder/decoder block this layer normalizes each word consisting of multiple vectors into a single Attention vector for the next decoder block or a linear layer.



Those analogies of machine translation English to French is also applied in our signal modulation scenario, just the input we change to signal data sequences. Every type of signal modulation has a different pattern of data, thus we can classify using the transformer network and take advantage of its modules.

Proposed Transformer Architecture

We proposed a new architecture that works specifically only to handle the noisy signal modulation / low SNR signal modulation as you can see in the figure below. We apply batch normalization to prevent overfitting and two fully connected networks along with Alpha Dropout. Alpha dropout has the advantage to keep the mean and variance of inputs to their original values, to ensure the self-normalizing property even after this dropout. For the activation function, we used SeLU that stands for Scaled Exponential Linear Unit to support the self normalizing property by the dropout layer. Last we keep using Lazy Adam as the model optimizer (the same with ResNet modified).

Optimizer : Lazy Adam

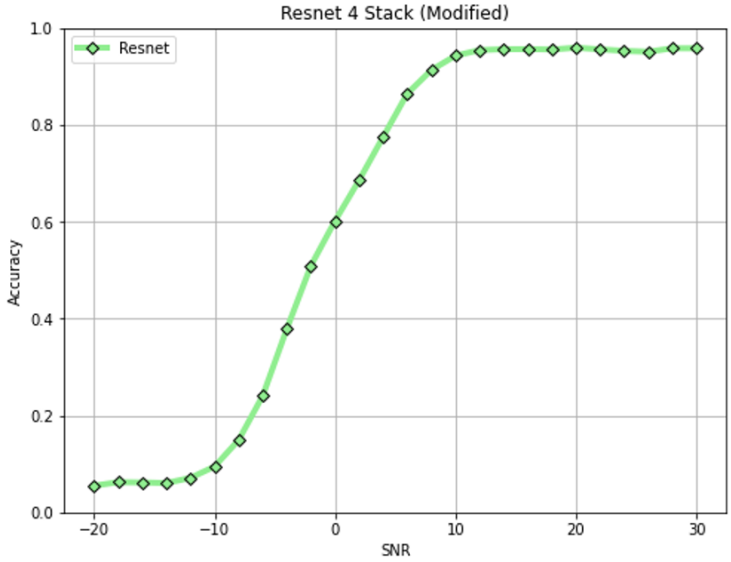
Layer	Output Dimension
Input	2 x 1024
Transformer Block Feed forward : 256 nodes	2 x 1024
Global Average Pooling	1024
Batch Normalization	1024
Alpha Dropout (0.3)	1024
FC/SeLU - Alpha Dropout (0.2)	128
FC/SeLU - Alpha Dropout (0.2)	128
FC/Softmax	24

VI. Result

- **Key Result High SNR (Modified Resnet)**

```
Overall Accuracy -20:  0.055106539309331376
Overall Accuracy -18:  0.062257511504105384
Overall Accuracy -16:  0.06172171088251218
Overall Accuracy -14:  0.06048918156161806
Overall Accuracy -12:  0.07119425480982357
Overall Accuracy -10:  0.09462177525142107
Overall Accuracy -8:  0.14932500688768482
Overall Accuracy -6:  0.24072169767224455
Overall Accuracy -4:  0.3788732394366197
Overall Accuracy -2:  0.5092386655260907
Overall Accuracy 0:  0.6015986919792897
Overall Accuracy 2:  0.6856315396113603
Overall Accuracy 4:  0.7753110733148331
Overall Accuracy 6:  0.8644160583941606
Overall Accuracy 8:  0.9129090440608091
Overall Accuracy 10:  0.9426221934768548
Overall Accuracy 12:  0.9540583356630323
Overall Accuracy 14:  0.9559833694866233
Overall Accuracy 16:  0.9560578661844484
Overall Accuracy 18:  0.9553463465192606
Overall Accuracy 20:  0.9590241145440844
Overall Accuracy 22:  0.9557135514819256
Overall Accuracy 24:  0.9525109170305677
Overall Accuracy 26:  0.9507161284583222
Overall Accuracy 28:  0.957817055393586
Overall Accuracy 30:  0.9578554778554779
```

Maximum Accuracy : 95.9 %
Baseline Max Accuracy : 93%



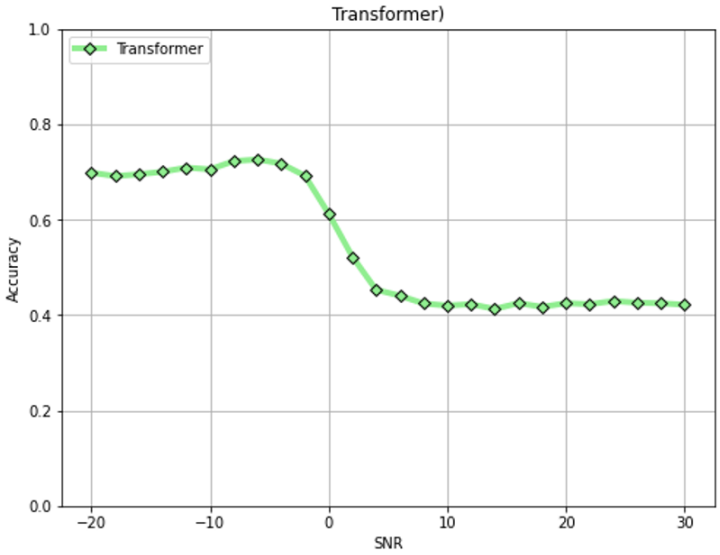
Result for High SNR Model

From the figure above, we could surpass the baseline maximum accuracy from 93.7% become 95.9% with our modified ResNet.

- **Key Result Low SNR (Transformer Network)**

```
Overall Accuracy -20:  0.6982902235861465
Overall Accuracy -18:  0.6911421911421911
Overall Accuracy -16:  0.6949346561098922
Overall Accuracy -14:  0.7002109898174479
Overall Accuracy -12:  0.708933982187127
Overall Accuracy -10:  0.7052417866371354
Overall Accuracy -8:  0.7226126885952582
Overall Accuracy -6:  0.726027397260274
Overall Accuracy -4:  0.7173520923520924
Overall Accuracy -2:  0.6917192920039136
Overall Accuracy 0:  0.6120804180320609
Overall Accuracy 2:  0.5221254987305042
Overall Accuracy 4:  0.45270022042615726
Overall Accuracy 6:  0.4405187835420394
Overall Accuracy 8:  0.42406798245614036
Overall Accuracy 10:  0.41975195222783646
Overall Accuracy 12:  0.4227296866702071
Overall Accuracy 14:  0.4125624553890079
Overall Accuracy 16:  0.42485574789169994
Overall Accuracy 18:  0.41695212999560827
Overall Accuracy 20:  0.4248495270851247
Overall Accuracy 22:  0.422582646459275
Overall Accuracy 24:  0.4292117057737938
Overall Accuracy 26:  0.4257234726688103
Overall Accuracy 28:  0.4245810055865922
Overall Accuracy 30:  0.4215823315627621
```

Maximum Accuracy : 72.6 %
Baseline Max Accuracy : 20%

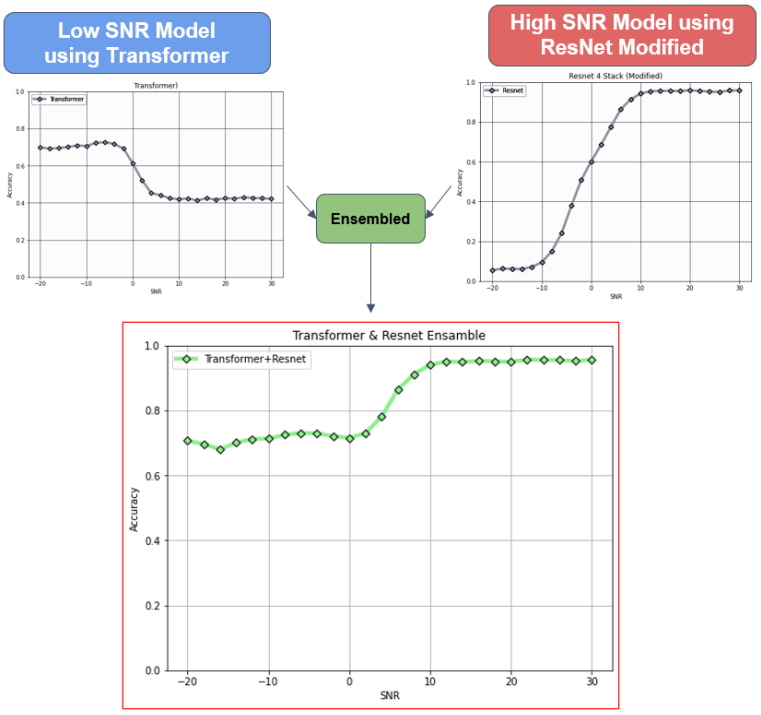


Result for Low SNR Model

From the figure above, we could recognize signal in Low SNR rate with by accuracy of 72.6%, the previous method even could not recognize signal with Low SNR (-20,0) rate and got maximum accuracy only 20%.

- **Key Result Low SNR + High SNR (Ensemble both Network)**

```
Overall Accuracy -20:  0.7062022984472323
Overall Accuracy -18:  0.6958828911253431
Overall Accuracy -16:  0.6800182898948332
Overall Accuracy -14:  0.7000183755972069
Overall Accuracy -12:  0.7118704714196261
Overall Accuracy -10:  0.7131408797013022
Overall Accuracy -8:  0.7245682249751334
Overall Accuracy -6:  0.7303317082496893
Overall Accuracy -4:  0.7290627227369921
Overall Accuracy -2:  0.7207435203769963
Overall Accuracy 0:  0.7150226244343891
Overall Accuracy 2:  0.729703494528768
Overall Accuracy 4:  0.7806192346332999
Overall Accuracy 6:  0.8631714733253478
Overall Accuracy 8:  0.9112794157424939
Overall Accuracy 10:  0.9399318630087861
Overall Accuracy 12:  0.9505992132467295
Overall Accuracy 14:  0.9485655927954694
Overall Accuracy 16:  0.9520711143695014
Overall Accuracy 18:  0.9508876272866541
Overall Accuracy 20:  0.9491071428571428
Overall Accuracy 22:  0.9561925365062196
Overall Accuracy 24:  0.955230575991387
Overall Accuracy 26:  0.9544656957360941
Overall Accuracy 28:  0.9519886615289219
Overall Accuracy 30:  0.9557119942969168
```

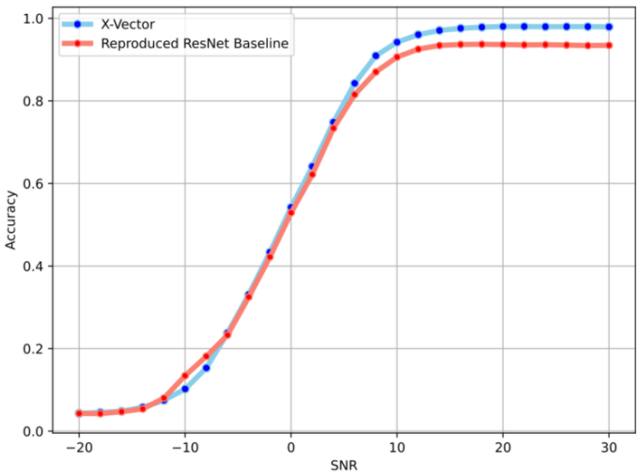


The result after ensemble two neural networks

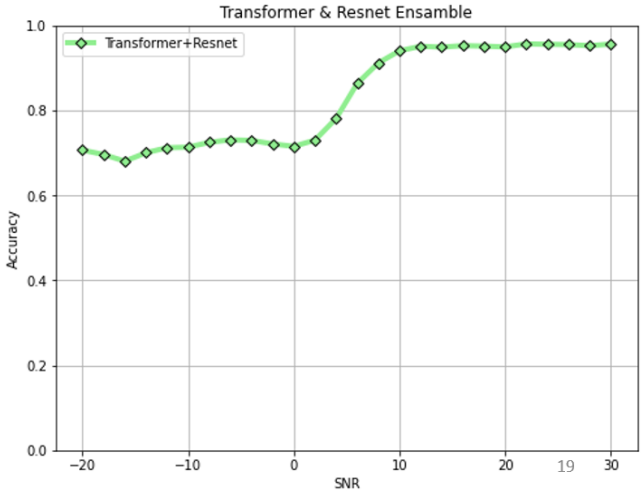
After ensemble two neural networks together, we could take advantage of both neural networks and we could get good results both for the signal with Low SNR and signal with High SNR.

- **Comparison with Baseline**

High SNR (maximum accuracy)
X-Vector : **98%**
Resnet Baseline : 93.7%
Resnet Modified (Ours) : 95 %



Low SNR (-20,0)
X-Vector : 17%
Resnet Baseline : 14%
Transformer (Ours) : **70.0 %**



Comparison with baseline, ours get better result both in Low SNR and High SNR

We got a better result for overall accuracy for Low SNR and High SNR. In high SNR, we got improvement from 93.7% to 95%. We also have a very significant improvement in Low SNR from only 14% to 70%. The transformer model could solve noisy signal from Low SNR Rate.

VII. Conclusion

1. We show that using an ensemble method our classifier can learn the signal not only for high SNR but also low SNR.
2. Transformer models can achieve up to 70% accuracy at low SNR values which is a big improvement in comparison with previous approaches.
3. Our modified ResNet gets higher accuracy than the baseline model and is comparable with state-of-the-art.
4. We successfully **make a new end-to-end deep learning model for modulation signal classification.**

VIII. References

[1] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-Air Deep Learning-Based Radio Signal Classification," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168-179, Feb. 2018, DOI: 10.1109/JSTSP.2018.2797022.

[2] Harper, Clayton A., et al. "Enhanced Automatic Modulation Classification using Deep Convolutional Latent Space Pooling." *ASILOMAR Conference on Signals, Systems, and Computers. 2020*.

[3] A. J. Uppal, M. Hegarty, W. Haftel, P. A. Sallee, H. Brown Cribbs, and H. H. Huang, "High-Performance Deep Learning Classification for Radio Signals," *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, 2019, pp. 1026-1029, DOI: 10.1109/IEEECONF44664.2019.9048897.

[4] S. Huang et al., "Automatic Modulation Classification Using Compressive Convolutional Neural Network," in IEEE Access, vol. 7, pp. 79636-79643, 2019, DOI: 10.1109/ACCESS.2019.2921988.

[5] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł. & Polosukhin, I. (2017), Attention is all you need, in 'Advances in Neural Information Processing Systems' , pp. 5998--6008 .