Deep Learning for Interference Identification: Band, Training SNR, and Sample Selection

Xiwen Zhang, Tolunay Seyfi, Shengtai Ju, Sharan Ramjee, Prof. Aly El Gamal, ECE Department, Purdue University

Prof. Yonina C. Eldar, Math and Computer Science Department, Weizmann Institute of Science

SPAWC, 2019

PURDUE
UNIVERSITY

Why Wireless Interference Identification?

Problem Setup

- dataset generated by Schmidt[a]
- 225,225 sample vectors for 15 classes in the SNR range of -20 dB to 20 dB with the step size of 2 dB
- Each sample vector consists of 128 I/Q samples, corresponding to 12.8µs
- I/Q samples of each sample vector are also transformed into the frequency domain by using the Fast Fourier Transform (FFT)

Class Index	Technology	Center Frequency	Channel Widtł
1	Bluetooth	2422 MHz	1 MHz
2	Bluetooth	2423 MHz	1 MHz
3	Bluetooth	2424 MHz	1 MHz
4	Bluetooth	2425 MHz	1 MHz
5	Bluetooth	2426 MHz	1 MHz
6	Bluetooth	2427 MHz	1 MHz
7	Bluetooth	2428 MHz	1 MHz
8	Bluetooth	2429 MHz	1 MHz
9	Bluetooth	2430 MHz	1 MHz
10	Bluetooth	2431 MHz	1 MHz
11	WiFi	2422 MHz	20 MHz
12	WiFi	2427 MHz	20 MHz
13	WiFi	2432 MHz	20 MHz
14	Zigbee	2425 MHz	2 MHz
15	Zigbee	2430 MHz	2 MHz

a M. Schmidt, D. Block, U. Meier. "Wireless Interference Identification with Convolutional Neural Networks".

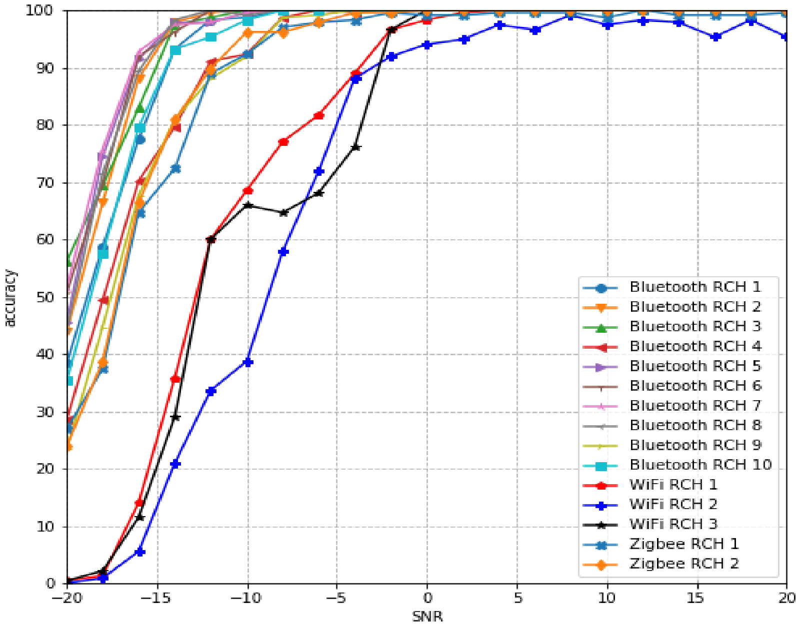
Architecture	Activation Function	Convolutional Layer	Dense Layer	Recurrent Cells	Residual Stacks	Accuracy
CNN [a]	ReLU, Softmax	64 $3*1$, 1024 $3*2$	$126976*128,\ 128*15$			0.8941
CNN	ReLU, Softmax	256 $3*1$, 256 $3*2$	$31744*1024,\ 1024*15$			0.8962
LSTM	ReLU, Softmax		$512*15$	512, 4		0.8965
ResNet	ReLU, Softmax		$128*128,\ 128*128,\ 128*15$		5	0.8938
CLDNN	ReLU, Softmax	256 $3*1$, 256 $3*2$	$512, 256,\ 256, 15$	256		0.8950

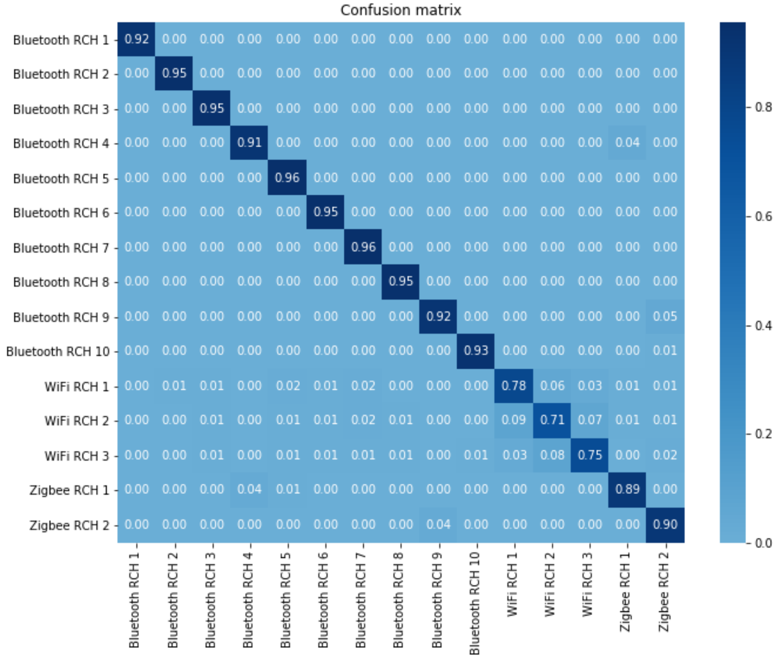
Previous Results and Improvement

- four different architectures are studied: CNN, ResNet, CLDNN, and LSTM
- based on FFT I/Q data
- our proposed CNN architecture delivers a slightly higher accuracy
- average training time we obtained for our proposed CNN architecture is around 108s, as opposed to a 180s training time obtained for the original CNN architecture

a M. Schmidt, D. Block, U. Meier. "Wireless Interference Identification with Convolutional Neural Networks".

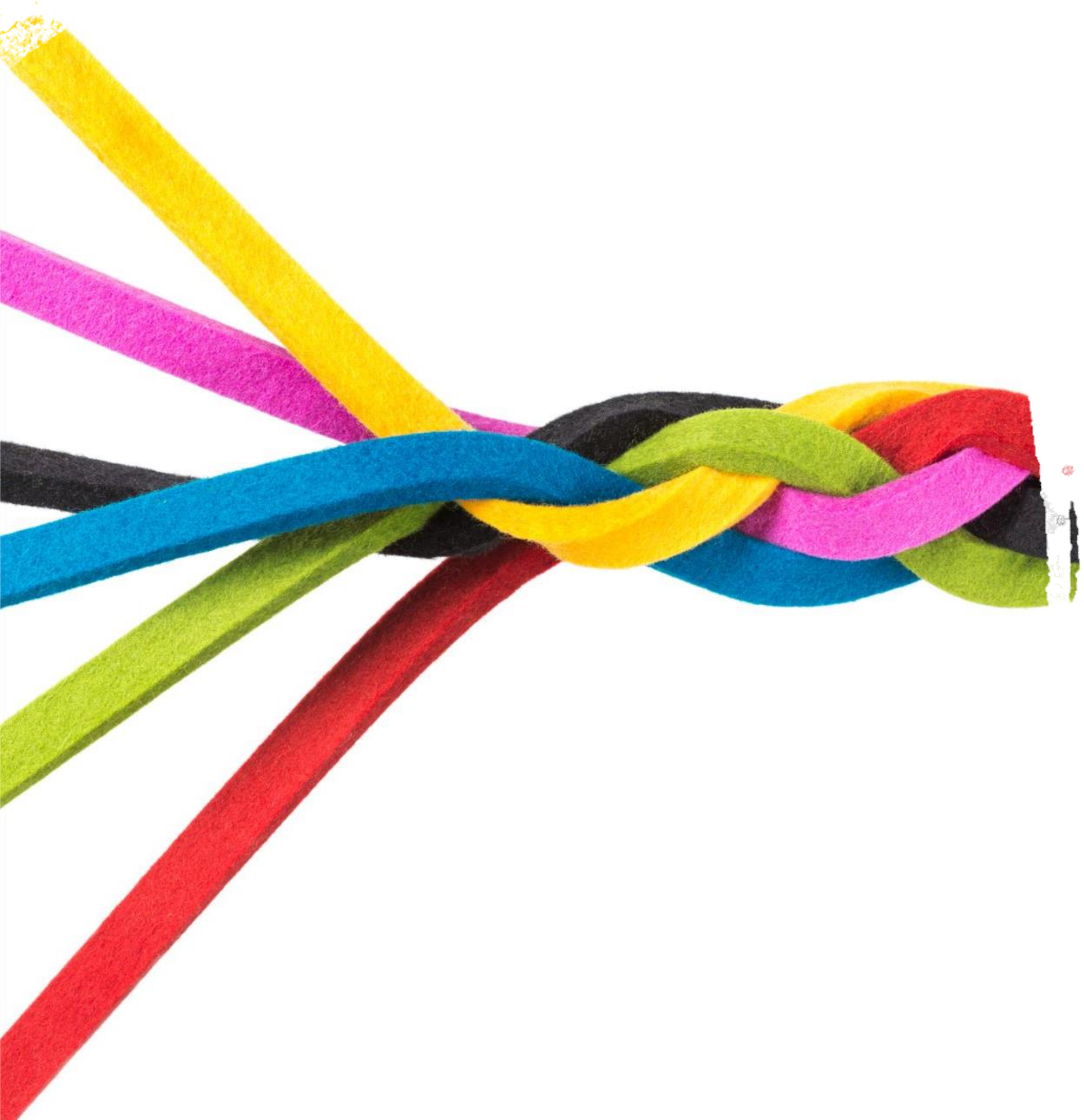
Results





Results

- Classification accuracies for the 3 classes of WiFi signals are significantly lower
- Focus on the confusion between different WiFi channels



Band Selection

Band Selection

Use only a **subset of the 10 MHz frequency range** to train and test the neural network classifiers

length of each sample is shorter, neural network shrinks correspondingly

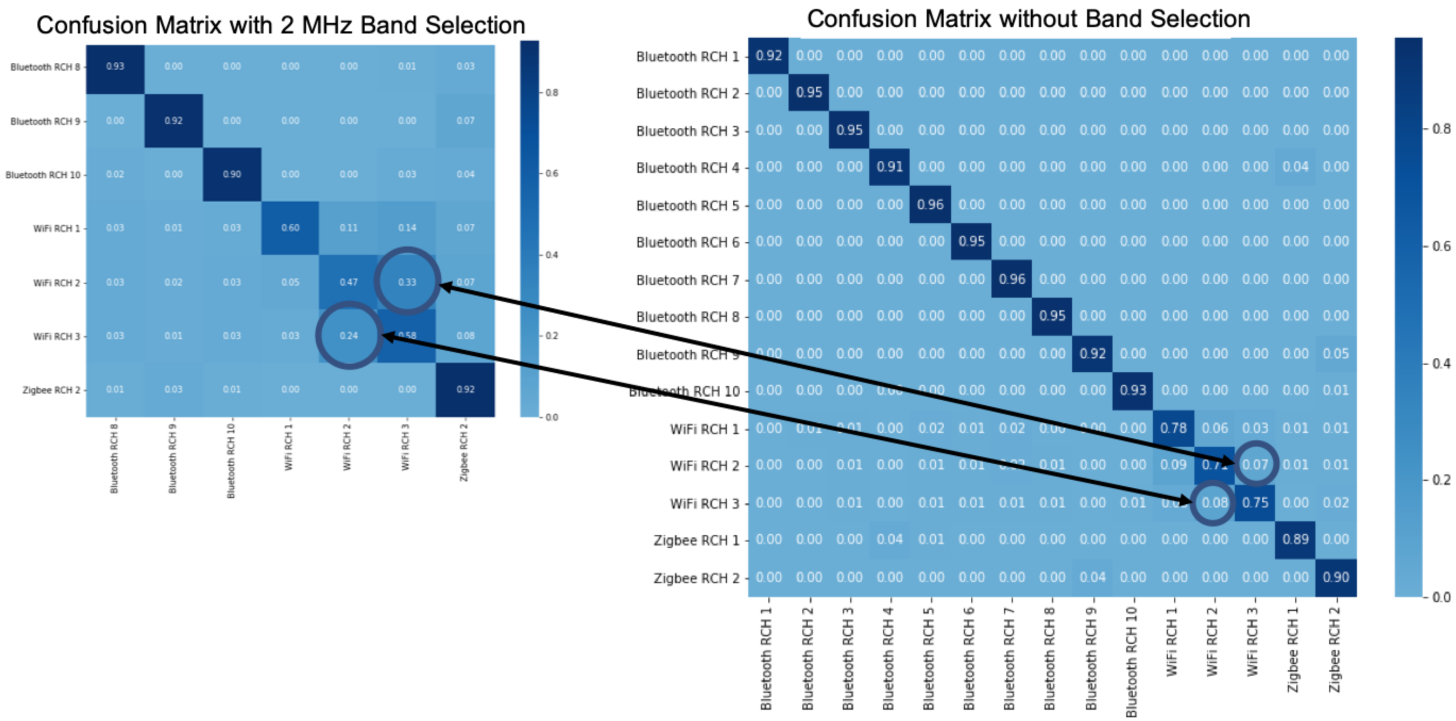
Band selection results in fewer classes, since not all are observable

Band Selection

- Start with selecting a narrow band of **2 MHz**: from **2429 MHz** to **2431 MHz**
- **7** observable classes: **3 bluetooth, 3 WiFi, 1 Zigbee**
- Training time is reduced by **more than 60%**
- Accuracy for **WiFi signals** is affected the most

	10 MHz	2 MHz
Bluetooth Accuracy	94.02%	91.49%
WiFi Accuracy	74.67%	52.55%
Zigbee Accuracy	89.18%	92.86%
Total Training Time	108.64s	40.75s

Band Selection: 2 MHz



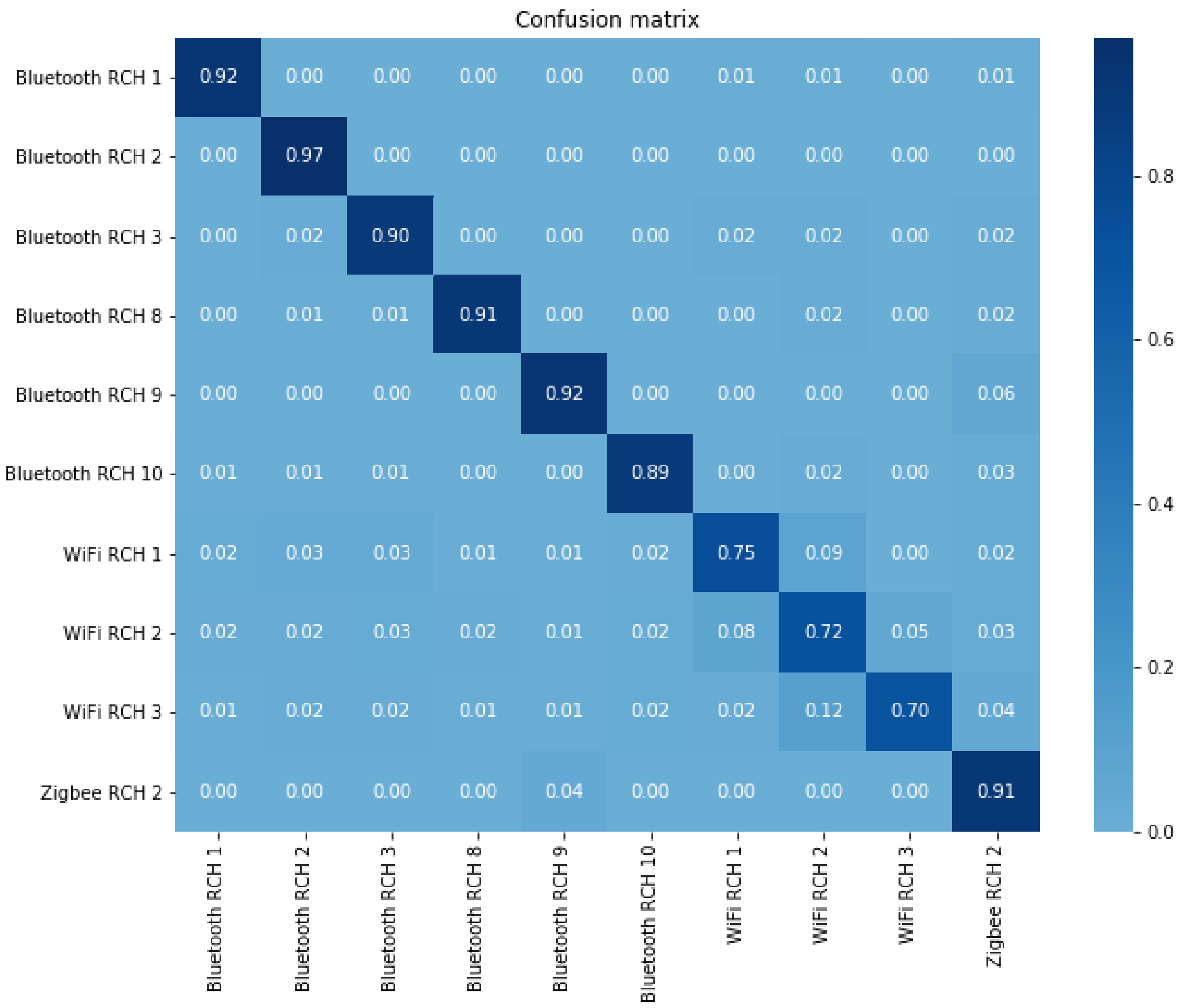
Select a narrow band of **2 MHz**: from **2429 MHz** to **2431 MHz**

Serious confusion between **WiFi RCH 1** and **WiFi RCH 2**

Select another narrow band to differentiate them!

Band Selection: 4 MHz

- Select 2 narrow bands: **2422-2424** MHz and **2429-2431** MHz
- **10** observable classes: **6 bluetooth, 3 WiFi, 1 Zigbee**



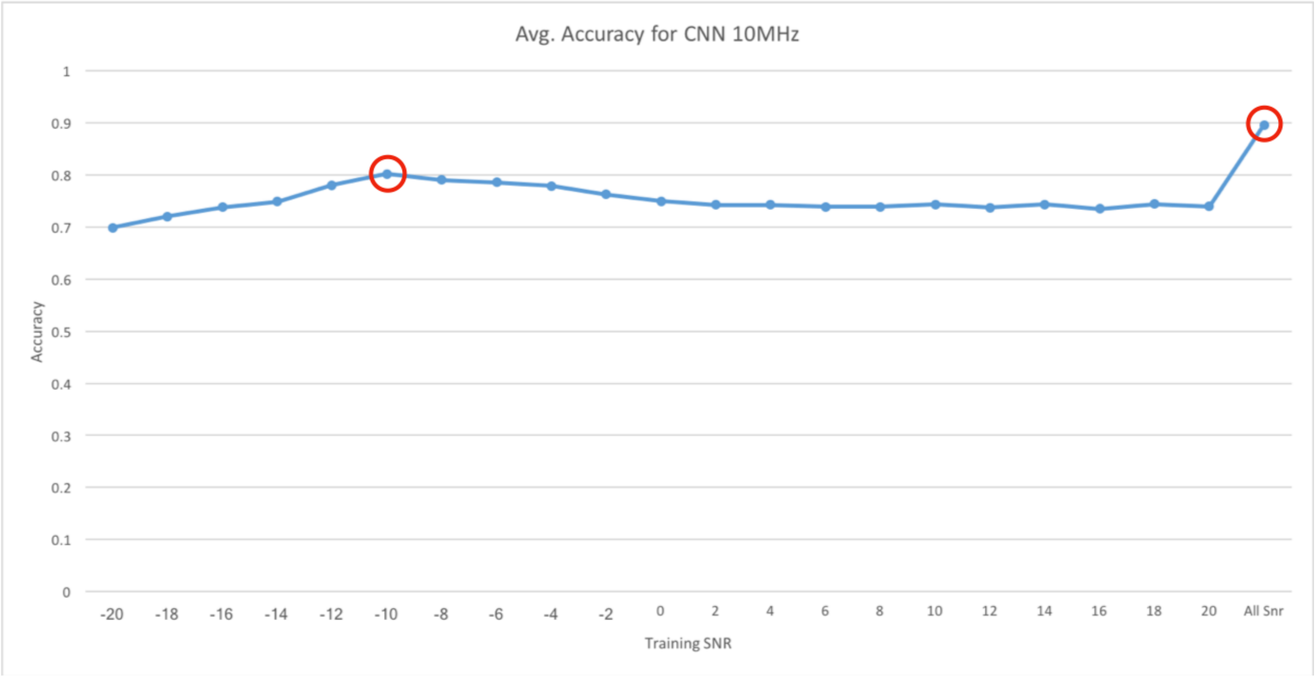
Band Selection

- Accuracy for **WiFi signals** is improved by **20%**
- 4 MHz band selection reduces the training time by **40%**
- Accuracy for every technology is preserved

	10 MHz	2 MHz	4 MHz
Bluetooth Accuracy	94.02%	91.49%	91.96%
WiFi Accuracy	74.67%	52.55%	73.23%
Zigbee Accuracy	89.18%	92.86%	89.67%
Total Training Time	108.64s	40.75s	60.10s

Training SNR Selection: 10 MHz dataset

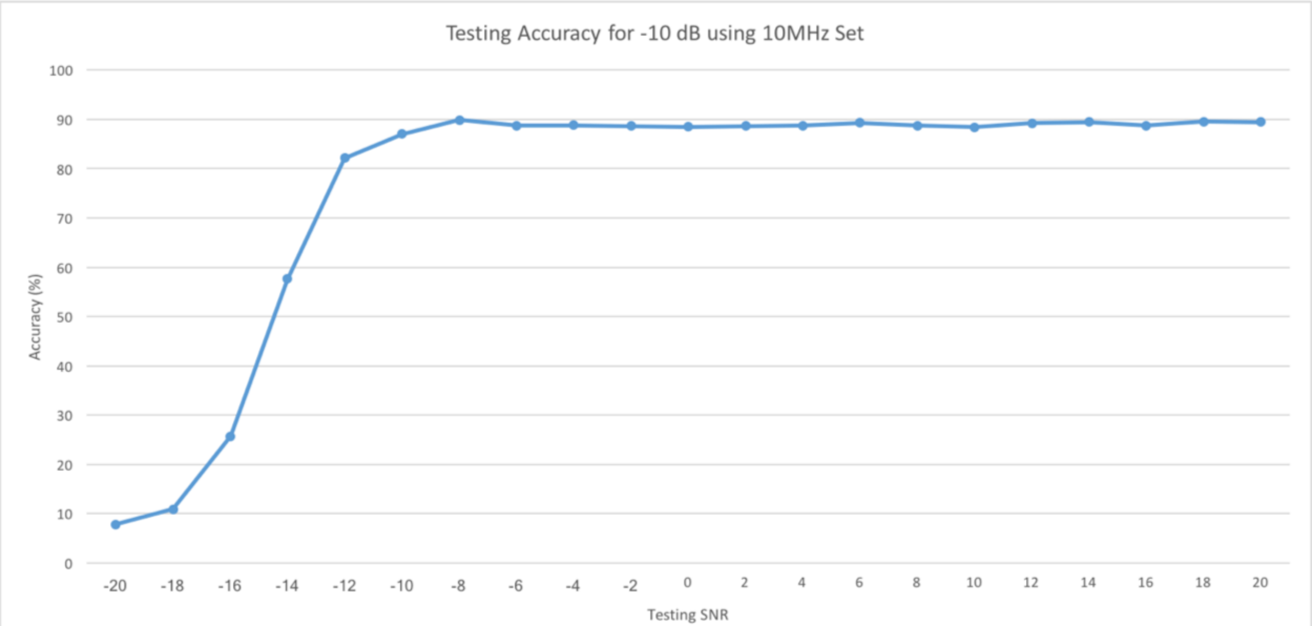
- Use data at a single SNR value to train the model
- Training time was reduced drastically
- High accuracy for high SNR values
- Testing accuracies for different training SNR values are close
- Training with -10 dB results in the best average accuracy of about 80%



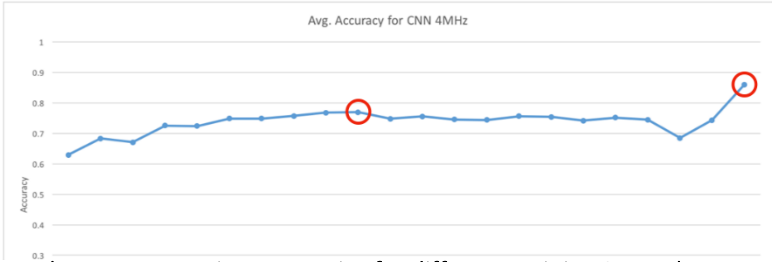
the average testing accuracies for different training SNR values

Training SNR Selection: 10 MHz dataset

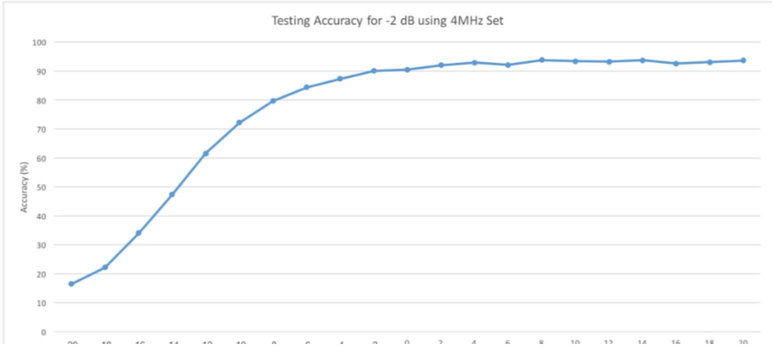
- Training time per epoch is reduced from **16.37s** to **0.984s**
- Total training time is reduced by **92.3%**



testing accuracy for each SNR value while training with -10 dB



the average testing accuracies for different training SNR values



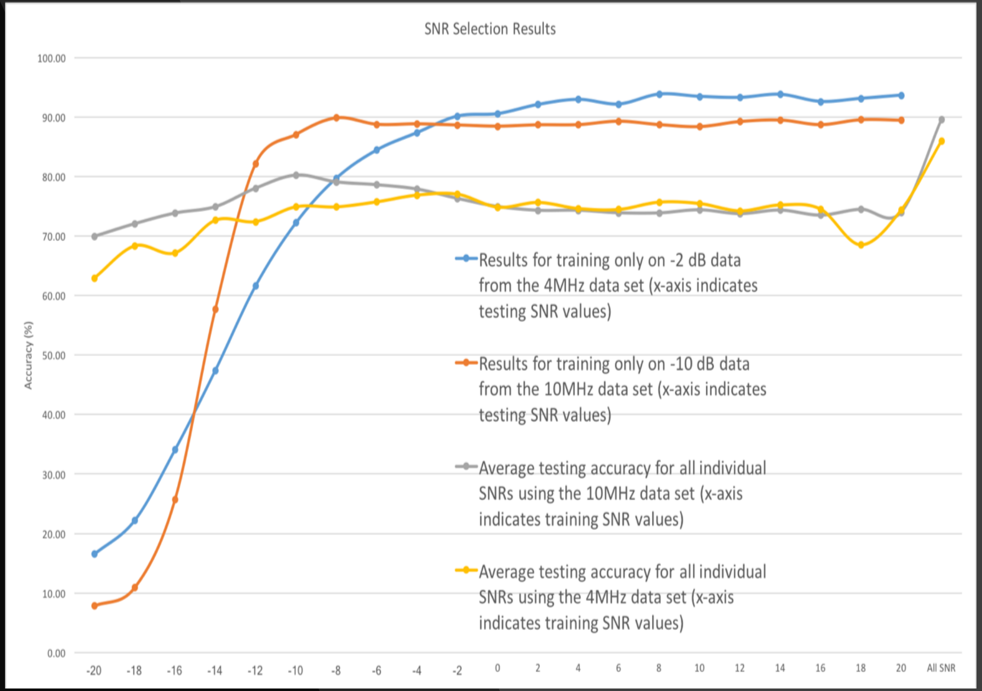
testing accuracy for each SNR value while training with -2 dB

Training SNR Selection: 4 MHz dataset

- Training with only -2 dB data led to best performance with an accuracy of **77%**
- Total training time is reduced by **90.9%**

SNR selection

With training SNR selection, the training time is drastically reduced, while the high accuracy for high SNR values is maintained

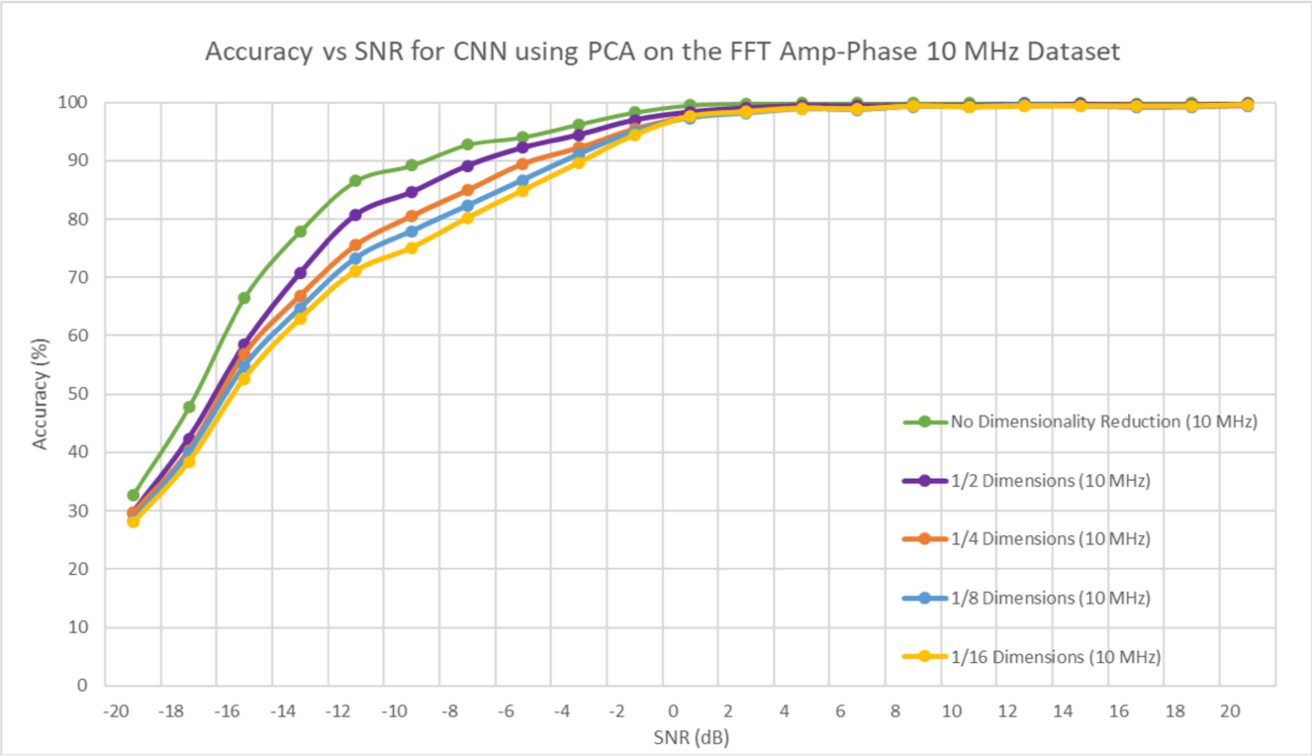


	Time per Epoch	Number of Epochs	Accuracy
All SNR 10 MHz	16.37s	6.6	0.8962
-10 dB 10 MHz	0.984s	8.5	0.8022
All SNR 4 MHz	4.12s	15.8	0.8614
-2 dB 4 MHz	0.61s	9.7	0.77

PCA and Sample Selection

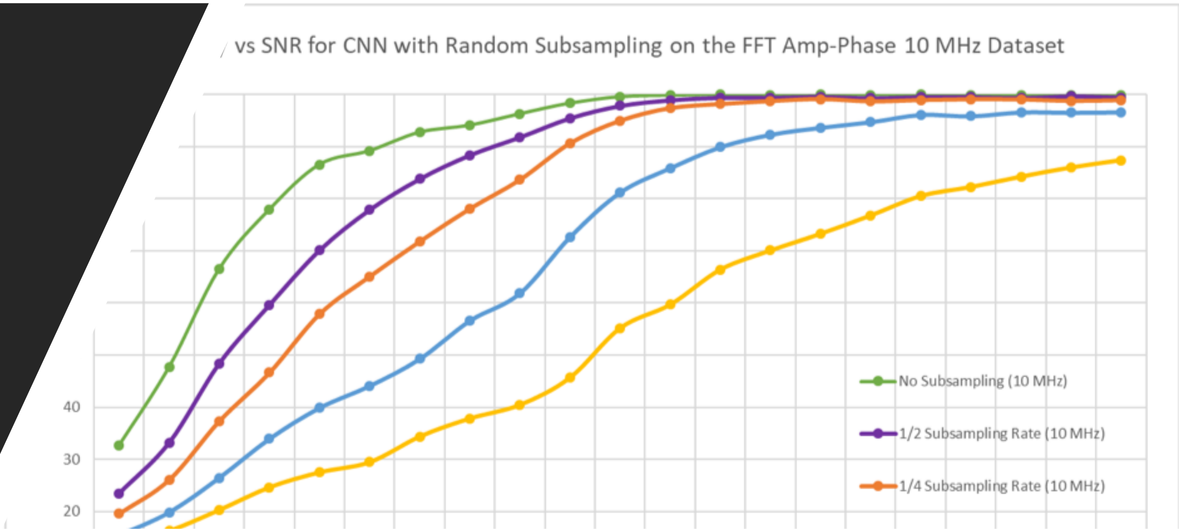
PCA and Sample Selection

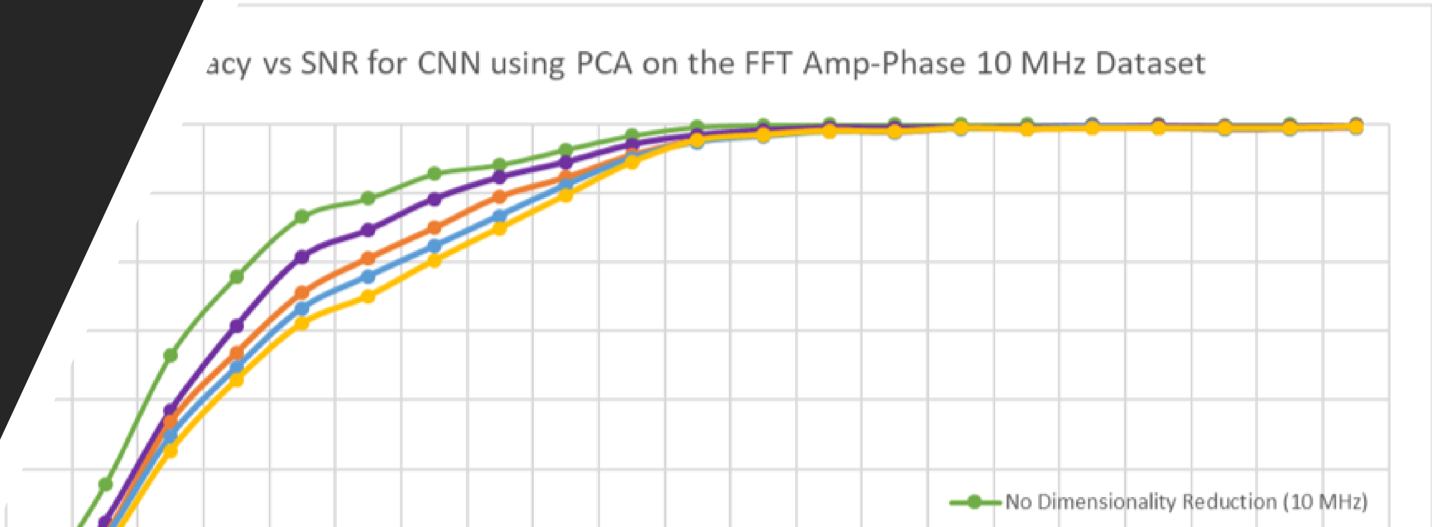
- Use **PCA** and various **subsampling** techniques to reduce the number of dimensions
- High accuracy for SNR values above 0 dB for a compression of **16x**
- Training time reduced by **87.97%**, average accuracy is **84.11%**



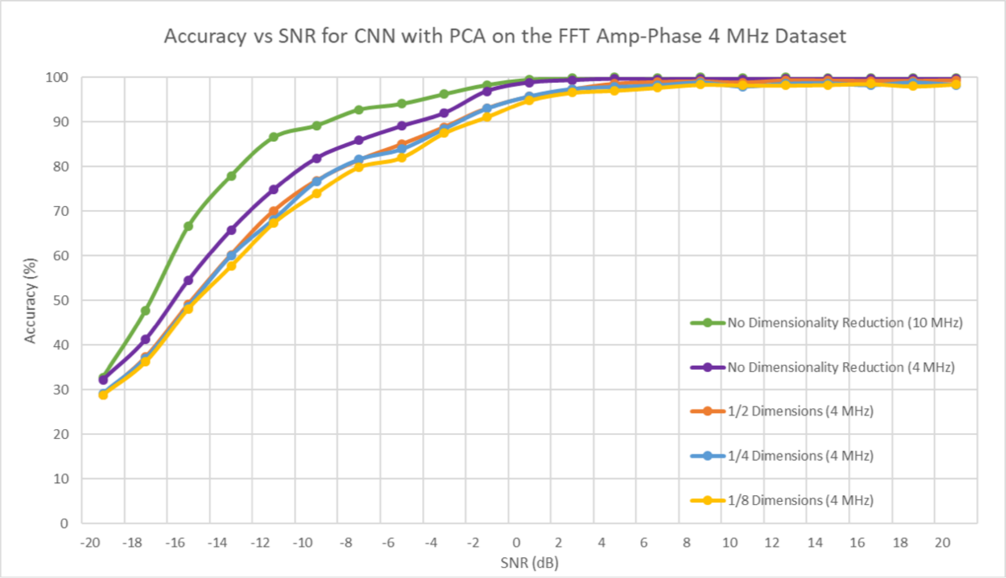
PCA and Sample Selection

- Random Subsampling results in large drops in accuracy at low SNR values compared to PCA
- High accuracy at high SNR values for a subsampling rate as low as 1/4
- Similar results with **Uniform Subsampling**





PCA and Sample Selection



- Combine **PCA** with **Band Selection** (Apply PCA on the 4 MHz Amp-Phase Dataset)
- Training time is reduced signicantly
- Classication accuracies at moderately high SNR values are still robust

PCA and Sample Selection

Dimensions/Samples	Time per Epoch	Epochs	Accuracy
All (10 MHz)	16.37s	6.6	0.8962
1/2 (10 MHz)	7.79s	8.6	0.8726
1/4 (10 MHz)	3.86s	8.5	0.8576
1/8 (10 MHz)	2.16s	7.4	0.8487
1/16 (10 MHz)	1.78s	7.3	0.8411
All (4 MHz)	4.12s	15.8	0.8614
1/2 (4 MHz)	2.72s	12.1	0.8358
1/4 (4 MHz)	1.64s	8.6	0.8310
1/8 (4 MHz)	1.33s	8.2	0.8220

- Number of features is reduced by PCA, while training time is reduced proportionally
- Signicant reduction of total training time (by about 90%)
- Classication performances are mostly preserved

Confidence-Based Ensemble Method

	Mean	Min	Max
CNN	89.764%	89.462%	89.952%
LSTM	89.713%	89.488%	89.972%
ResNet	89.405%	89.126%	89.701%
CLDNN	89.903%	89.704%	90.041%

- Considered network architectures: CNN, LSTM, CLDNN & ResNet
- All models result in similar accuracies of **89.xx%** on the test set

Confidence-Based Ensemble Method

Ensemble method combines decisions from multiple models to improve the overall performance

The simplest ensemble method is voting

We tried voting, it doesn't work well…

Instead, for every decision made by each model, we assign **a confidence score** for it

To predict the label for each sample in the test set, we choose **the most confident model** to make the decision

There are two candidate confidence scores:

- The precision score
- The output of the last layer (softmax)

Confidence-Based Ensemble Method

	MEAN	MIN	MAX
CNN	89.764%	89.462%	89.952%
LSTM	89.713%	89.488%	89.972%
ResNet	89.405%	89.126%	89.701%
CLDNN	89.903%	89.704%	90.041%
Softmax-based	90.067%	89.921%	90.312%
Precision-based	89.743%	89.519%	89.941%

Confidence-Based Ensemble Method

SNR	(# times) softmax is better	(# times) precision is better	(# times) softmax and precision are equal
-20	22	3	0
-18	24	1	0
-16	25	0	0
-14	25	0	0
-12	23	1	1
-10	24	1	0
-8	20	4	1
-6	23	2	0
-4	20	4	1
-2	16	9	0
0	12	9	4
2	12	6	7
4	8	8	9
6	2	16	7
8	7	5	13
10	7	13	5
12	4	14	7
14	9	10	6
16	2	13	10
18	4	14	7
20	1	20	4

Confidence-Based Ensemble Method

	Mean	Min	Max
CNN	89.764%	89.462%	89.952%
LSTM	89.713%	89.488%	89.972%
ResNet	89.405%	89.126%	89.701%
CLDNN	89.903%	89.704%	90.041%
Softmax-based	90.067%	89.921%	90.312%
Precision-based	89.743%	89.519%	89.941%
combine both	90.081%	89.921%	90.339%

- Combining these two confidence measure:
  - Use softmax for SNR from –20 dB to 2 dB
  - Use precision score for SNR from 4 dB to 20 dB

What is Ax?

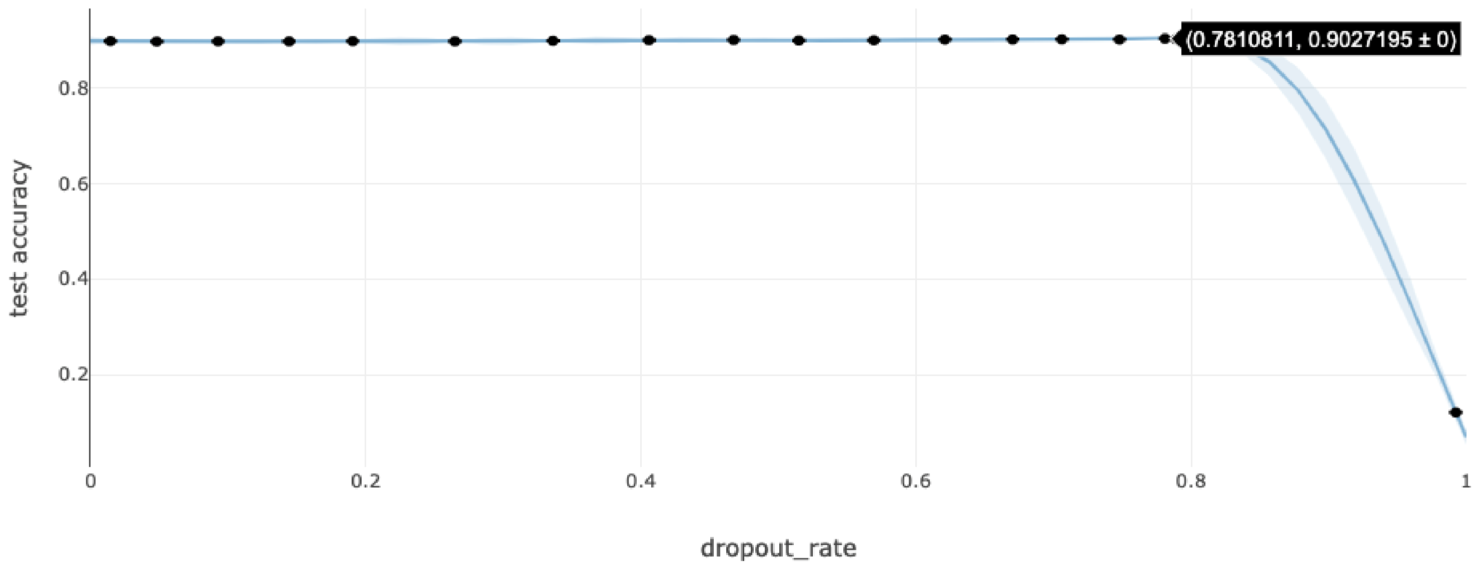
- **Ax** is a platform developed by Facebook to explore a large parameter space in order to identify optimal configurations in a resource-efficient manner
- It supports **Bayesian optimization** for continuous-valued configurations and bandit optimization for discrete configurations
- We can use it to tune hyper parameters for deep learning models

Ax to search for dropout rate

- Dropout is an effective technique for regularization
- In the previous work by Schmidt, they set the dropout rate to **0.6**
- It is a magic number. It is possible to use trial and error to determine the value
- We use Ax to search for the best value

Layer type	Input size	Parameters	Activation fct.
Convolutional layer	128×2	3×1 filter kernel 64 feature maps	Rectified linear
Convolutional layer	$64 \times 126 \times 2$	3×2 filter kernel 1024 feature maps Dropout 60 %	Rectified linear
Dense layer	126976×1	128 neurons Dropout 60 %	Rectified linear
Dense layer	128×1	15 neurons	Softmax

Ax to search for dropout rate



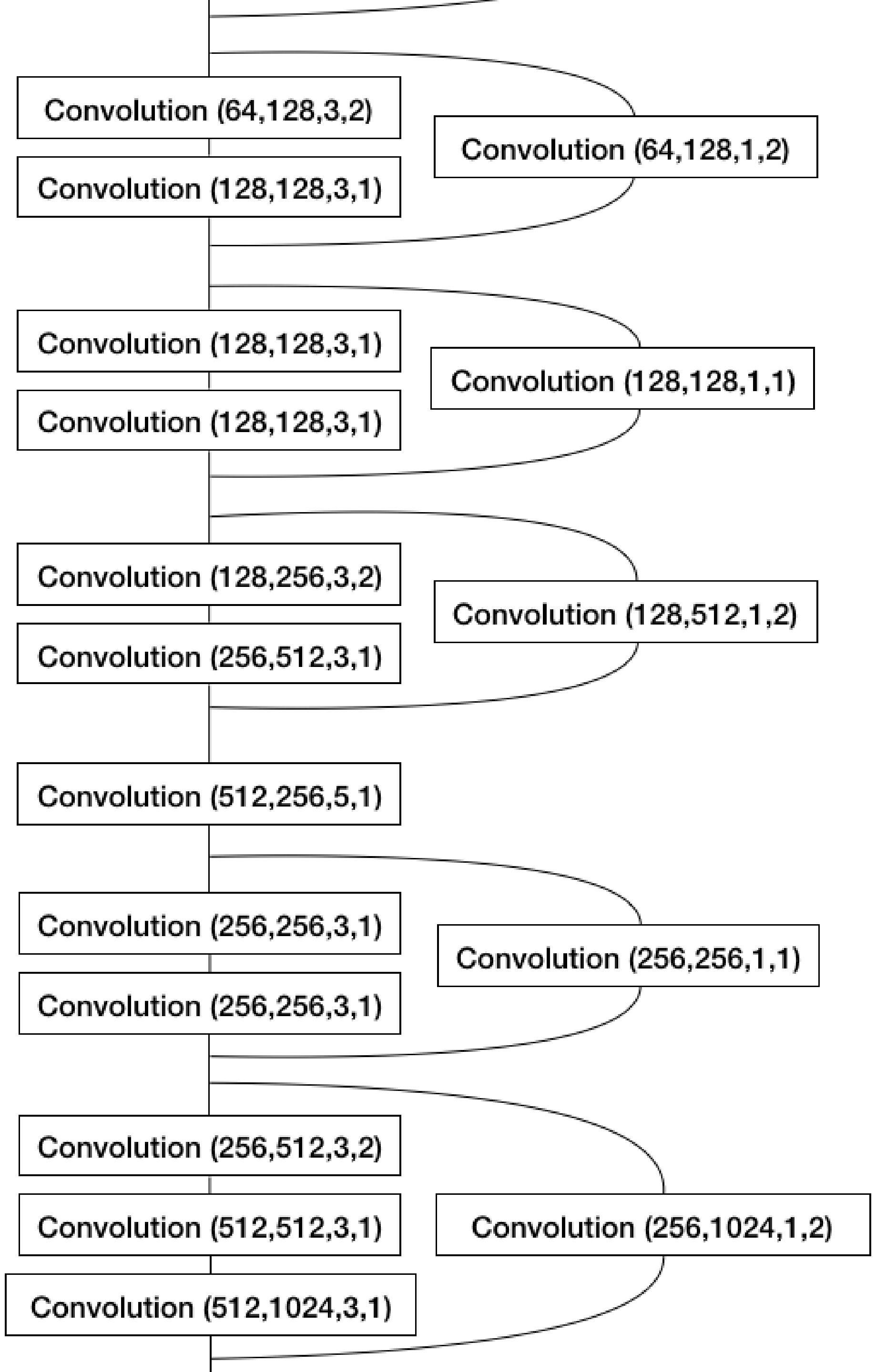
- The best dropout rate found by Ax is **0.78108**
- The accuracy on the test set is improved from **89.62%** to **90.27%**

Future Work & Challenges

- We plan to use Ax to optimize more hyperparameters, like **learning rate**, **number of filters**, **filter size**, **number of neurons in the fully connected layer**...
- Ax prefers models with **high variance**, which means it tends to look for models that **overfit** the data
- More hyperparameters are optimized, more experiments are needed for Bayesian Optimization algorithm to converge

What is AutoKeras?

- AutoKeras is an open source software library for automated machine learning (AutoML) or Neural Architecture Search (NAS)
- It is developed by DATA Lab at Texas A&M University
- It use network morphism guided by Bayesian optimization to search the best neural network architecture
- It’s more computation efficient compared with other NAS algorithms
- NASNet by Google takes 48000 GPU hours, which is unaffordable



AutoKeras to Search for Neural Network Architectures

- Architecture was found by AutoKeras after a 24-hour search
- It is a variant of ResNet
- Its accuracy on the test set is **90.22%**