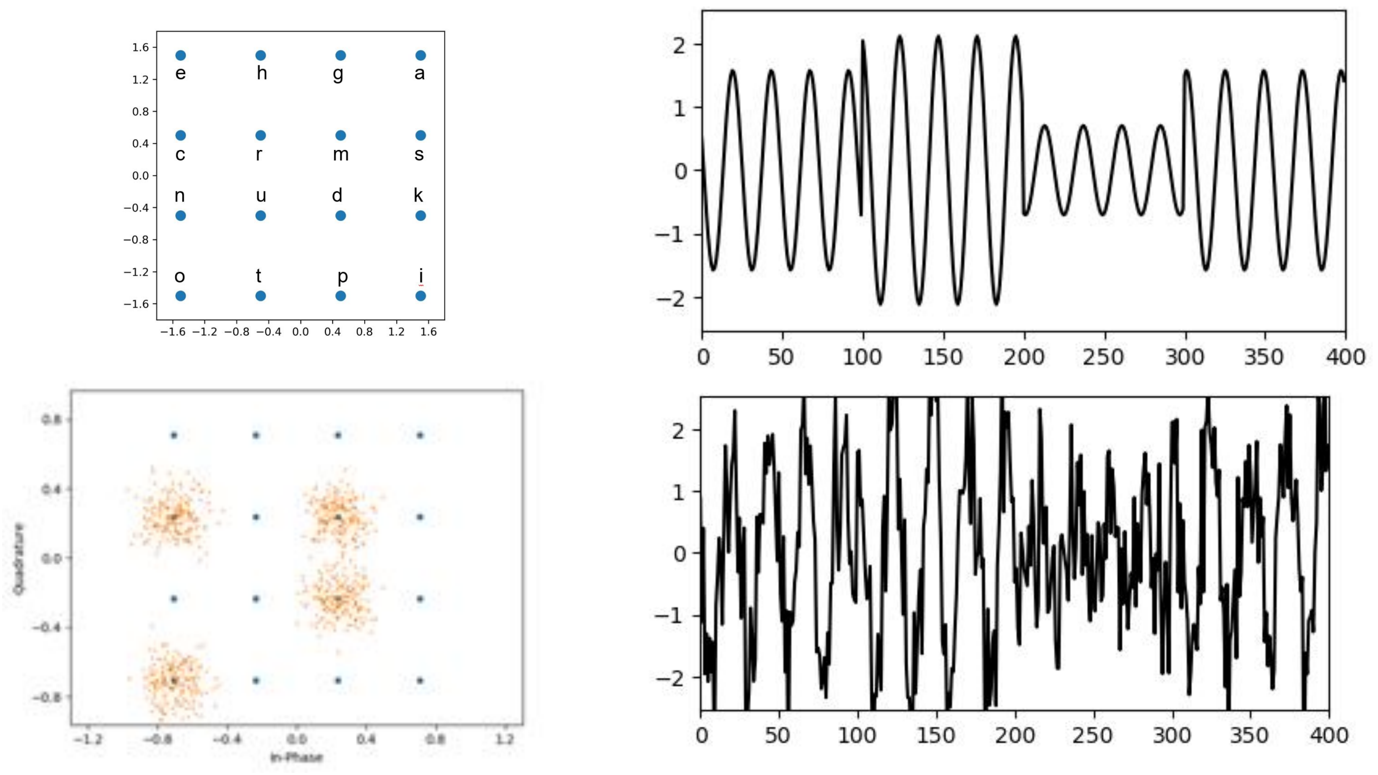
CS221 Interpretation of Noisy QAM Signals (bingdong, yueheng, smaccabe)

Video link: https://www.youtube.com/watch?v=qnRHYHihvqQ

Introduction

- Radio frequency (RF) signals transmit information by controlling the amplitude and phase of a carrier sine wave
- During process and transmission, these RF signals have noise added, creating errors at the output.of the receiver
- To interpret incoming signals, this noise must be overcome
- Digital RF signals are sequences of discrete symbols
- We looked at Quadrature Amplitude Modulation (QAM) signals:
  - Autoencoder technology to clean up noisy signals
  - Convolutional Neural Network to classify signals

16-QAM Signals and Noise



- Quadrature Amplitude Modulation signals can be represented by an X-Y ‘Constellation’ of points centered on grid locations
- In the time domain, signals are sections of sine waves with relative amplitude and phase determined by grid points
- Gaussian noise spreads QAM points to a cloud around center
- The noise distorts the time domain signal from sine portions

Our Signal Generation and Processing

- We assigned the most common English letters to QAM points
- And defined the signals for 20 words created from those letters
- We wrote code to generate signals and plots in Python
- Then we made multiple signal instances for each word
  - With varying levels of noise
  - And transformed to images by plotting the time-domain signals
- We trained a CNN to classify samples of all 20 signals
- We trained an Autoencoder to de-noise samples of 3 signals

References

[1] Leslie Casas, Attila Klimmek, Nassir Navab, Vasileios Belagiannis, “Adversarial Signal Denoising with Encoder-Decoder Networks”, arXiv:1812.08555v1

[2] Dibakar Sil, Arindam Dutta, Aniruddha Chandra, "CNN based noise classification and denoising of images", IEEE Dataport, 2019

[3] Diederik P. Kingma, Jimmy Lei Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION”,ICLR 2015

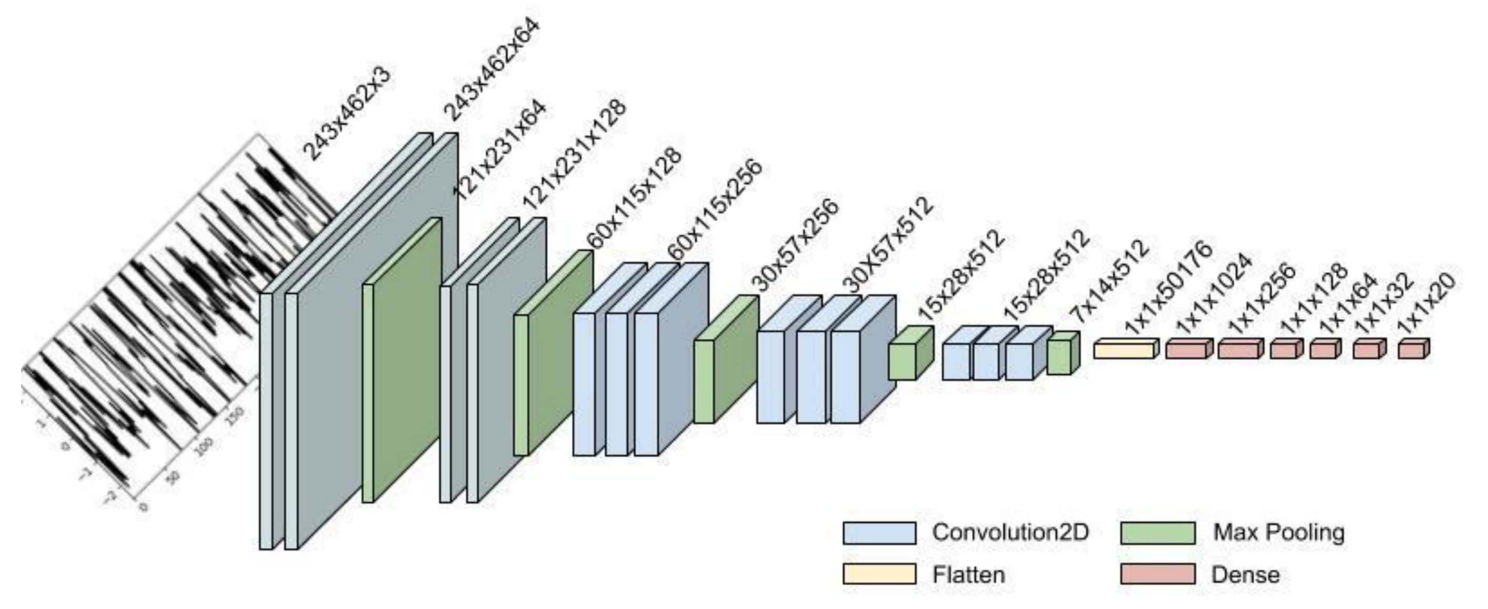
[4] O’Shea, et. al, “Over the Air Deep Learning Based Radio Signal Classification”, arXiv:1712.04578v1

[5] Joel Akeret, et. al, “Radio frequency interference mitigation using deep convolutional neural networks”, arXiv:1609.09077v2

[6] https://en.wikipedia.org/wiki/Quadrature_amplitude_modulation

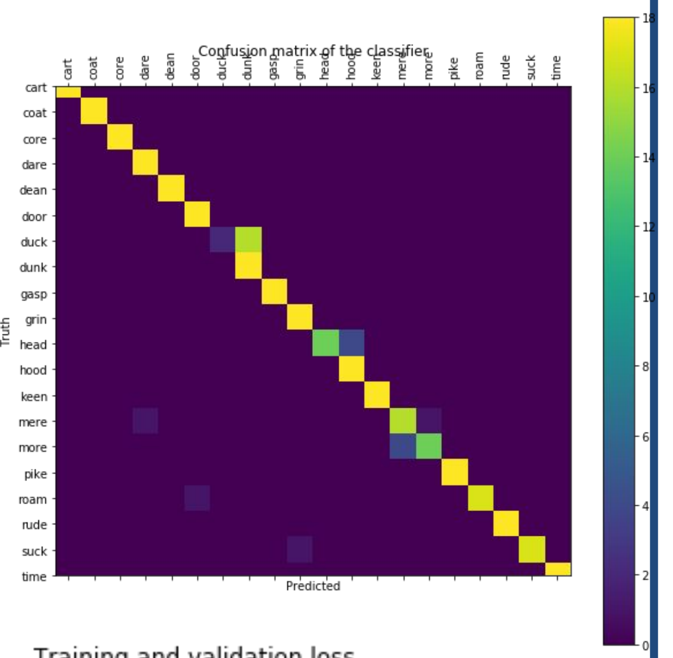
Convolutional Neural Network Architecture

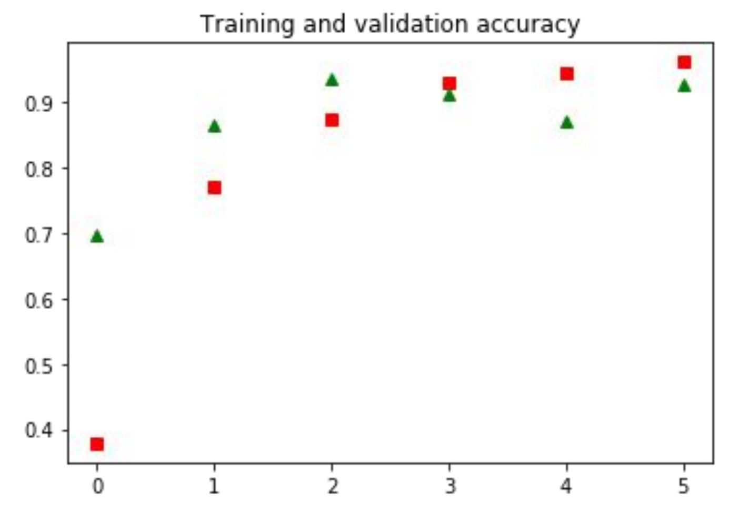
- VGG-like Convolutional Neural Network
- Input: 243x462 greyscale images with various levels of noise
- Output: Classification of the images in one of 20 classes
- Loss function: mean square error for all pixel depth values

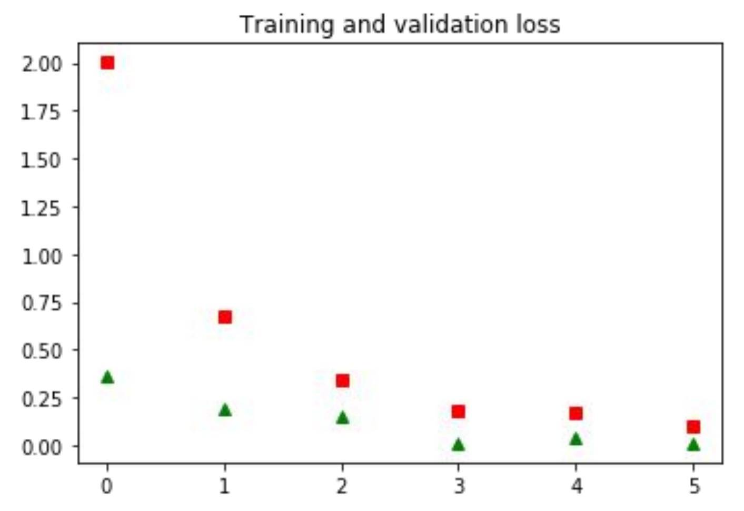


Experiments and Implementations

- Implemented in Keras with Tensorflow backend
- Leveraged VGG-16 with transfer learning
- Images are shuffled before training
- Images pixel values are rescaled to (0, 1)
- Images are split to 60% training data, 25% validation data, 15% test data
- Experimented with different noisy level
  - Include noisy level from 2 to infinity
  - Only include noisy level from 2 to 50
- Hyperparameters experiments, including:
  - Batch size
  - Dropout rate
  - Learning rate
  - Adam optimization decay rate
  - Number of layers
  - Number of Neurons in each layer







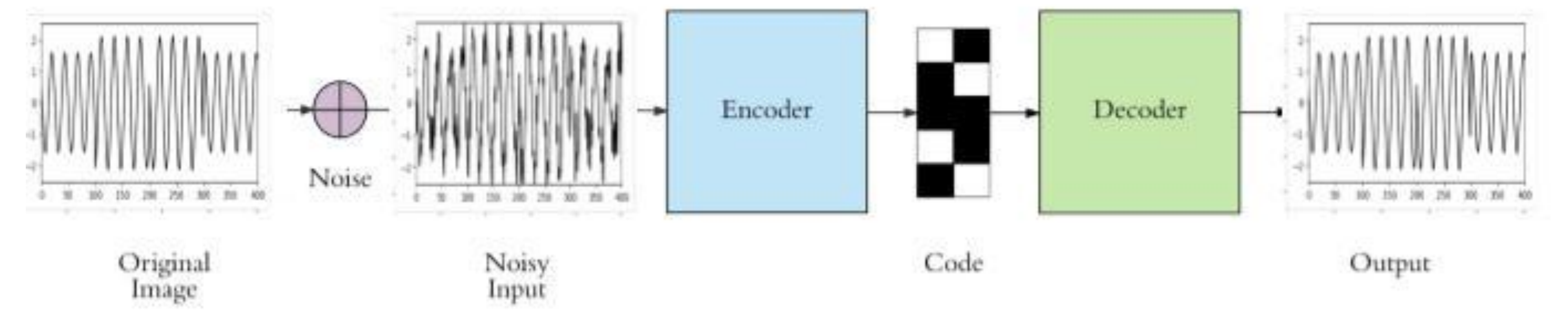
Confusion matrix, Training (Red) and Validation (Green) accuray, loss of one experiment that yields good accuracy. Parameters: Number of layers and neurons in each layer see architecture, batch size 2, dropout rate: 0.001, learning rate: 0.0001, adam optimization decay rate: 0.000001, noise levels used correspond to Signal to Noise Power ratios ranging from 2 (high noise) to 50 (medium)

Conclusions and Future Work

- Concl: A CNN can classify noisy 16-QAM signals with high accuracy
- Concl: An autoencoder can clean up very noisy QAM signals well
- Future: Experiments with less constrained inputs
- Future: Experiments with multiple different modulations at in the data
- Future: Experiments where amplitude and phase noise are not equal

Autoencoder Architecture

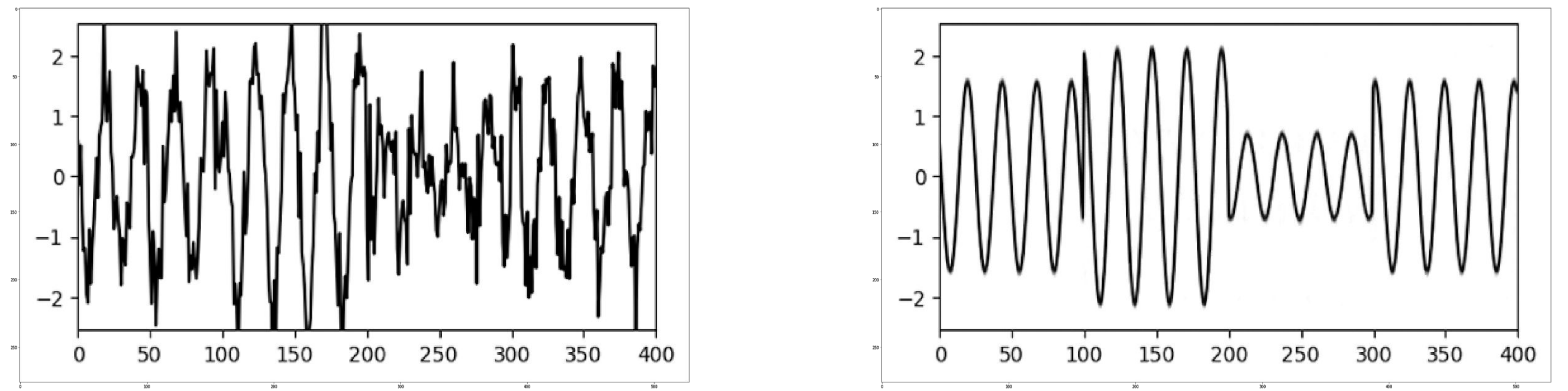
- Keras standard Autoencoder architecture
- Input: 243x462 grayscale image with various levels of noise
- output: 243x462 grayscale image with noise removed
- 3 different image class: Cart, Rude, Coat
- 180 training image, 45 test image



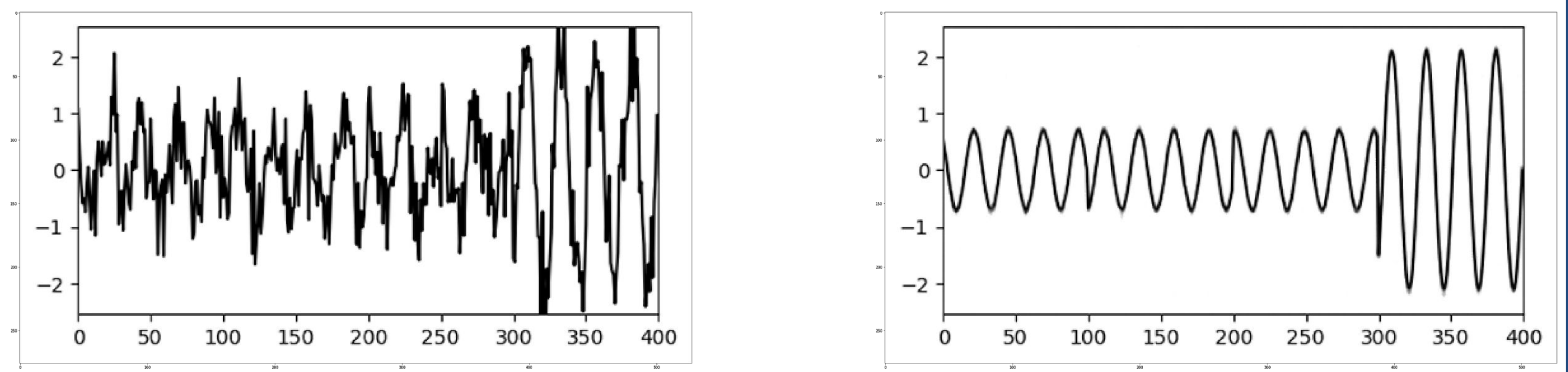
Encoder and Decoder process flow

Experiments and Implementations

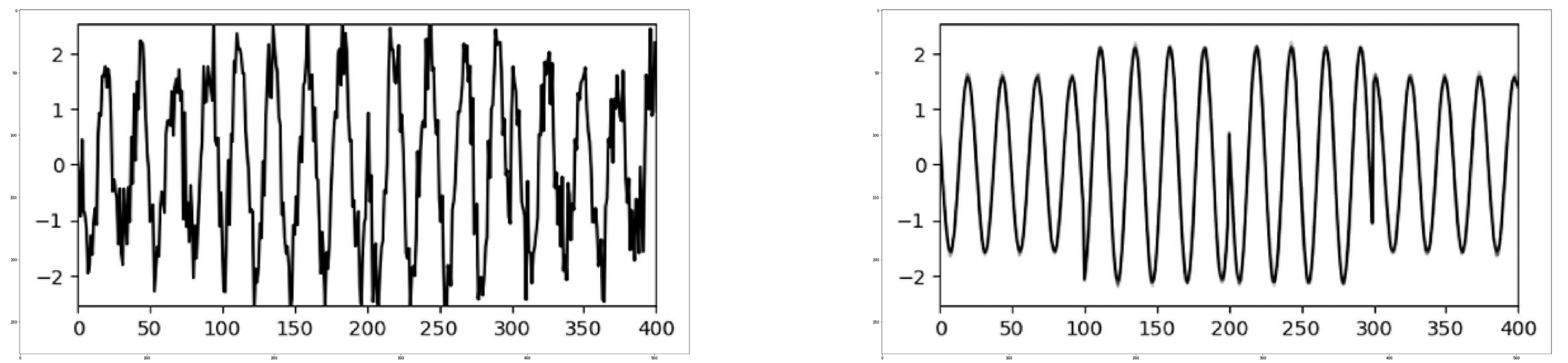
- Implemented in Keras with Tensorflow backend
- Images are ordered according to class for autoencoder training
- Modified stride size, kernel size and batch size in hyperparameter tuning for faster training
- Chose MSE as loss function and ADAM as optimizer
- Original noisy image and reconstructed clean images are shown side by side below
- Validation loss went from 0.22 (epoch 1) to 0.001 (epoch 100)



Class label Cart before and after Autoencoder



Class label Rude before and after Autoencode



Class Label Coat before and after Autoencoder