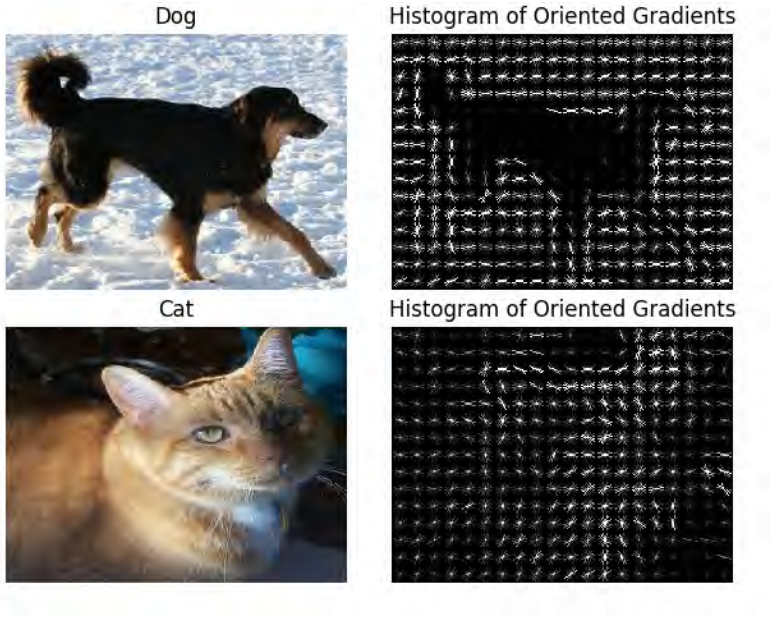
Chapter 2: Introduction to Deep Learning and Applications in Time Series Prediction

In this chapter, we are going to explore a little bit about deep learning. Deep learning is an advanced form of machine learning. One of the most significant differences between deep learning and machine learning is the former's model size is typically much larger, and there are much more parameters needed to be trained. Another crucial difference is deep learning could automatically learn the relevant features from the training data to make a prediction, compared to a typical machine learning model where we need to hand engineer the features from the training data first and then feed them into the model. Therefore, before the invention of deep learning, a lot of effort was spent in machine learning research on feature engineering, not the design of models.

For example, when we use SVM to perform an image classification task to tell whether an image is a dog or cat, we need to first handcraft some features and extract those features from the images. After that, the original images are no longer required. Features could be the gradient information of an image as we have demonstrated in the below image, where we extracted the gradient information from a dog image and a cat image. If we want to train an SVM classifier, we will need to convert all our dog images and cat images by the same feature transformations before feeding them to train the SVM.

Figure 8

EXAMPLE OF IMAGE RECOGNITION WITH SVM CLASSIFIER



However, if we want to solve the same problem by using deep learning, we simply need to feed the whole image into the neural networks, and the networks will automatically learn what kind of features are important to tell whether it is a dog or cat.

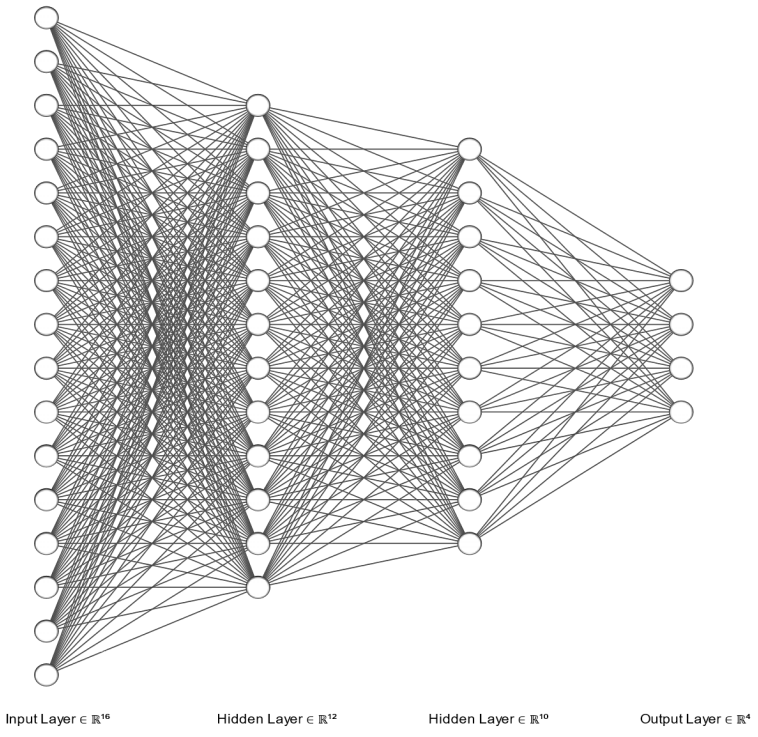
2.1 DEEP NEURAL NETWORKS (DNN)

Deep Neural Networks (DNNs) gained much attention since Alex Krizhevsky successfully applied them and won the ImageNet Large Scale Visual Recognition Challenge in 2012. A deep neural network will look similar to Figure 9, where we have a Multiple Layer Perceptron (MLP), and each node in the network is a neuron except the first input layer. The input layer is the training data, and the output layers are the results we want to predict. Between the input and output layers, there are multiple hidden layers. In each layer other than the input layer, the nodes are called neurons, which will use a non-linear activation function to map the input to the output. One of the reasons why deep neural networks are so powerful is that they typically have a lot of hidden layers and a massive number of neurons. This allows them to learn a very high dimensional relationship between input and output, which could be an ideal fit for financial data since they can seem entirely random sometimes.

The lines between neurons are called connections, which represent how the data flows between different neurons. In the case of an MLP, each neuron has a connection to all the neurons in the previous layer, and we call those dense layers or fully connected layers. Taking the above dog and cat image classification problem as an example, to use an MLP as a classifier, we would need to flatten the 2-D image into a 1-D vector and would only have one neuron at the end of the MLP, which will output the labels indicating dog or cat. As we have already pointed out, there is no process for extracting features from images.

Figure 9

ILLUSTRATION OF A DEEP LEARNING MODEL



Each circular node represents a neuron, and the lines denote the connectivity between neurons. Neurons will apply a weight matrix $\boldsymbol{W}$ and a bias term $\boldsymbol{b}$ to the input data $\boldsymbol{X}$ it receives and will then apply an activation function σ to the results before passing the results $\boldsymbol{Y}$ to the neurons in the next layer. This whole process could be summarized as

$$Y = \sigma(\boldsymbol{W}\boldsymbol{X} + \boldsymbol{b})$$

The activation function controls whether the current neuron should be active or not and how much it should be active. The range of the output of activation functions usually is in the range of $[-1,1]$ or $[0,1]$ and the choice of activation functions is usually a non-linear function such as a sigmoid function, Rectified Linear Unit (ReLU), or tanh function. The use of non-linear activations is one of the reasons the DNN models are capable of learning the non-linear relationships within the data and, thus, have better ability to represent the training data.[2]

Like machine learning, deep learning can be grouped into supervised learning, unsupervised learning, and reinforcement learning. For insurance applications, there is a recent success story from AXA Japan about applying deep learning in pricing.[3] AXA Japan's R&D team developed a deep learning model to predict significant claims, i.e., by entering the information of individual policies, we want to predict the likelihood that they will result in a substantial claim in the future. AXA entered over 70 input features including age, region, annual insurance premium, age of car, etc. into the deep learning model. The model is a fully connected neural network with three hidden layers. AXA used data in Google Compute Engine to train the model and Cloud Machine Learning Engine's HyperTune feature to tune hyperparameters. The resulting accuracy rate was about 78%.

2.2 DNN EXAMPLE OF TIME SERIES PREDICTION

In this section, we are going to demonstrate how to use deep neural networks (DNNs) to predict bitcoin prices. The DNN we are using in this section is mostly a feed-forward network. We are going to use two types of DNNs: Multi-layer Perceptron (MLP) and Convolutional Neural Networks (CNN).

2.2.1 MULTI-LAYER PERCEPTRON (MLP) VS. CONVOLUTIONAL NEURAL NETWORKS (CNN)

MLPs are probably one of the oldest DNN models and we have shown their network structure above in Figure 9. In an MLP, every neuron of the current layer will have one connection of every other neuron of the next layer. That is why the layers of an MLP are also called a fully connected or dense layer.

[2] Hornik, Kurt; Stinchcombe, Maxwell; White, Halbert. "Multilayer feedforward networks are universal approximators." University of California, San Diego . March 1989. https://www.sciencedirect.com/science/article/pii/0893608089900208

[3] Sato, Kaz. "Using Machine Learning for Insurance Pricing Optimization." March 2017. Google Cloud Platform https://cloud.google.com/blog/products/gcp/using-machine-learning-for-insurance-pricing-optimization

CNNs were invented to process 2-D images. However, in recent years, they are also widely used in time series data such as natural language processing or human activity recognition from accelerometer data recorded by smartphones or other wearable devices.[4]

It is natural to think that MLPs should be useful for time series applications. In some sense, MLPs are a lot like the traditional regression problem where we have a model to regress the input data. The difference is there is no need to design the mathematical model by hand; we only need to define how many neurons we need in the MLP, and the weights of the connections between different neurons describe the mathematical relationship to regress the training data. Nevertheless, the problem with MLPs is that the connections are so dense that it is very computationally demanding and slow, and it is also quite easy to overfit the training data.

As for a 1-D CNN, it is designed to find the pattern of a fixed segment of time series data. For each to-be-found pattern, we define a set of parameters, and we can predefine how many patterns we would like to reveal from each segment by experience. Therefore, CNNs require fewer connections between neurons across layers.

In the below experiment, we will present how to use both MLP and CNN in time series application with bitcoin history data.

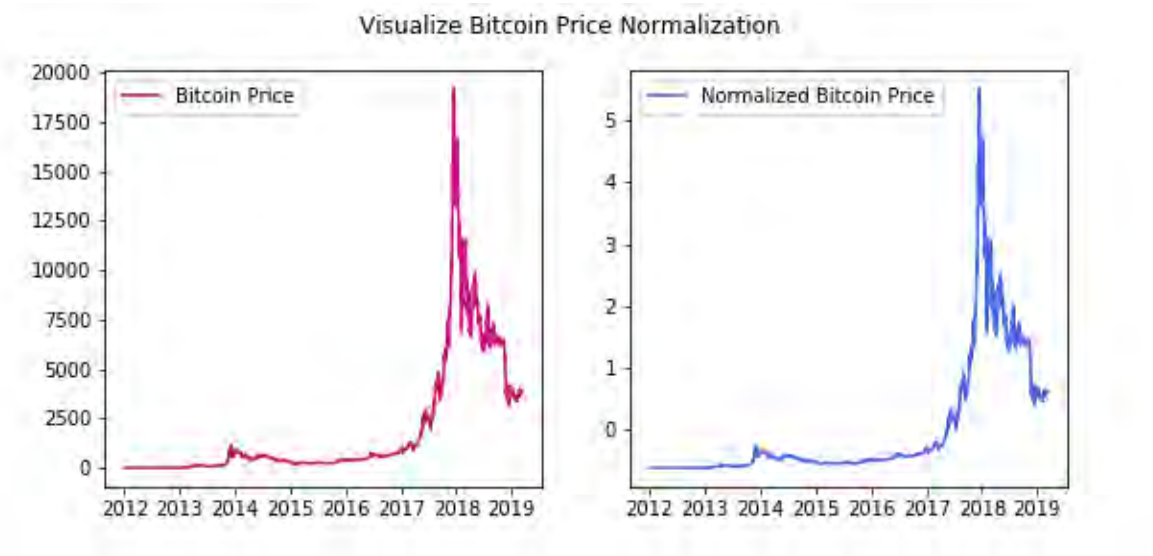
2.2.2 DATA PREPARATION

The dataset used is the bitcoin historical data from January 2012 to March 2019 and can be downloaded from the Kaggle website: https://www.kaggle.com/mczielinski/bitcoin-historical-data. The original dataset contains the bitcoin history data updated every minute with Open, High, Low, and Close prices. In this simple demonstration, we average the coin prices observed each day and use both an MLP and CNN to predict the next five days of bitcoin prices using the past 15 days prices.

There are 2,627 valid data points from January 2012 to March 2019. The original data is illustrated in the left figure below. It can be observed that the price is relatively flat between 2012 and 2017 and then rises quite high at the beginning of 2018 before falling later. The original data needs to be normalized first so that the model will treat every section of the data equally. Otherwise, the section with a large scale will have a bigger impact on the model and the model might overfit to those sections.

[4] Ackermann, Nils. "Introduction to 1D Convolutional Neural Networks in Kera for Time Sequences." Goodaudience, The Medium Spet 2018. https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf

Figure 10
BITCOIN PRICE NORMALIZATION



The time series bitcoin data does not explicitly have the ground truth label y (ground truth refers to the target variable being measured in training or testing), and both CNNs and MLPs are supervised, learning models. Therefore, we need to generate ground truth labels. As the goal here is to use the last 15 days of bitcoin prices to predict the following five days, both the MLP and CNN are used to try to find a mapping function $f(\cdot)$, which will perform the below mapping:

$\hat{y} = f(X[t_0 \cdots t_{14}])$, and the ground truth $y = X[t_{15} \cdots t_{19}]$ and the goal is to make sure the mean square error $\frac{1}{N}\Sigma\|y - \hat{y}\|^2$ is within some acceptable range. The 2,627 samples are split into two parts: the first 70% is used as training data, and the remaining 30% as testing data to evaluate the model accuracy after the training is complete.

2.2.3 CREATING THE MODEL

The MLP model used in this task consists of multiple dense layers (or fully connected layers), where each neuron in the previous layer will have a connection to every neuron in the next layer. Due to the use of dense layers, there are many parameters to be trained in the MLP model. In the MLP model we built, there are 21,509 parameters to be trained.

For the CNN model, multiple 1-D convolutional layers are used. Inside a convolutional layer, a set of weights (usually called filters or kernels) is systemically applied across the input data. For 2-D image detection, the filter is a 2-D dimensional array of weights. The filter is designed to detect the specific features of the input data, and the output of this operation is called a "feature map." Each convolutional layer will learn specific patterns from the data, and the next layer of the convolutional layer will learn more high dimension patterns from earlier layers. A Max-pooling layer is added right after the last convolutional layer to avoid overfitting. A dense layer is also used to make sure the output dimension is five. CNN models are quite effective at identifying the repeated patterns in the time series data; therefore, the required number of parameters is significantly smaller than for an MLP. In the CNN model we built, there are only ~9,000 parameters. For more details, please refer to the model on the GitHub site:

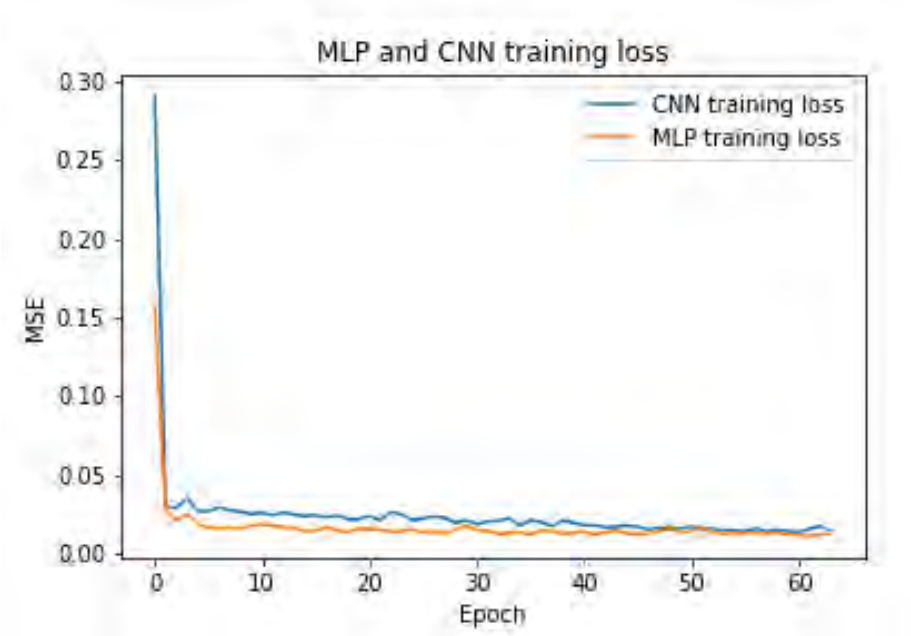
https://github.com/rranxxi/soa_research_ai_time_series/tree/master/bitcoin_cnn.

2.2.4 TRAINING THE MODEL

Both the CNN and MLP models were trained for around 64 epochs (an epoch is one complete presentation of the dataset to be learned to a learning machine). The MSE after each epoch of training is shown in Figure 15 for both models. The MSE of both models plateaued after around 20 epochs, which indicates that both of the models converged to their optimal state. Overall, MLPs tends to have slightly smaller MSEs than CNN models during the training stage. Next, we want to visit the testing results of the two models.

Figure 11

TRAINING LOSS COMPARISON OF MLP AND CNN MODELS IN BITCOIN TIME SERIES PREDICTION



2.2.5 RESULTS ANALYSIS

After the models were trained, we used them to predict bitcoin prices. We have a sliding window, which includes the 15 days of bitcoin data used both as the model input and predictor of the bitcoin price for the following five days. Then, we moved this sliding window by five days to predict the next five days. It is not surprising that, in the first 1,828 days, the predictions from both of the models have almost no errors since this part of the data has been used as training data. What is interesting is that, even with the testing data section, the predictions from both of the models are excellent. The MSE of the CNN model is 0.039 and the MSE of the MLP model is 0.0479, although the CNN model uses less than half of the parameters required by the MLP model. This result demonstrates that CNNs should be the preferred model in this time series analysis. This is, in fact, not counterintuitive. We human beings are also trying to recognize the patterns from the time series data to help us predict the future, and CNNs excel in identifying the pattern in both 1-D sequential data and 2-D image data. The fact that CNNs are proficient in recognizing pattern sets is a great advantage in time series predictions.

Figure 12
CNN/MLP PREDICTION ERROR

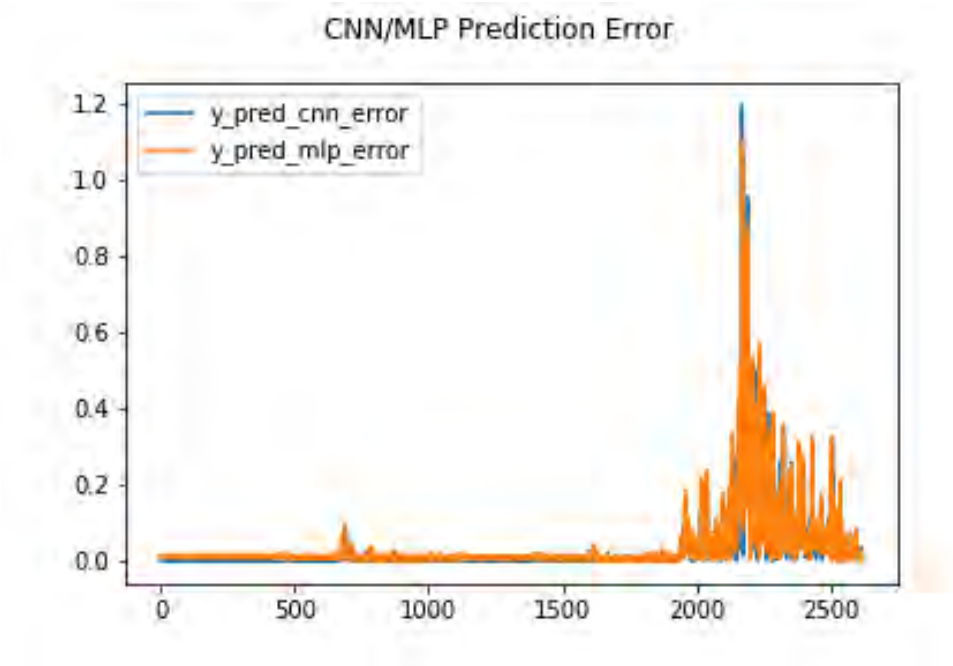


Figure 13

CNN/MLP PREDICTION VS GROUND TRUTH