WORKING PAPER · NO. 2023-100

Financial Machine Learning

Bryan T. Kelly and Dacheng Xiu

JULY 2023

stock-level panel, available directly from the Wharton Research Data Services server. Jensen *et al.* (2021) construct 153 stock signals, and provide source code and documentation so that users can easily inspect, analyze, and modify empirical choices. Furthermore, their data is available not just for US stocks, but for stocks in 93 countries around the world, and is updated regularly to reflect annual CRSP-Compustat data releases. These resources may be accessed at jkpfactors.com. Jensen *et al.* (2021) emphasize a homogenous approach to signal construction by attempting to make consistent decisions in how different CRSP-Compustat data items are used in various signals. Chen and Zimmermann (2021) also post code and data for US stocks at openassetpricing.com.

In terms of standardized data for aggregate market return prediction, Welch and Goyal (2008) post updated monthly and quarterly returns and predictor variables for the aggregate US stock market.[3] Rapach and Zhou (2022) provide the latest review of additional market predictors.

3.2 Experimental Design

The process of estimating and selecting among many models is central to the machine learning definition given above. Naturally, selecting a best performing model according to in-sample (or training sample) fit exaggerates model performance since increasing model parameterization mechanically improves in-sample fit. Sufficiently large models will fit the training data exactly. Once model selection becomes part of the research process, we can no longer rely on in-sample performance to evaluate models.

Common approaches for model selection are based on information criteria or cross-validation. An information criterion like Akaike (AIC) or Bayes/Schwarz (BIC) allows researchers to select among models based on training sample performance by introducing a probability-theoretical performance penalty related to the number of model parameters. This serves as a counterbalance to the mechanical improvement in fit due to heavier parameterization. Information criteria aim to select a model from the candidate set that is likely to have the best out-of-sample

[3]Available at sites.google.com/view/agoyal145.

prediction performance according to some metric.

Cross-validation has the same goal as AIC and BIC, but approaches the problem in a more data-driven way. It compares models based on their "pseudo"-out-of-sample performance. Cross-validation separates the observations into one set used for training and another (the pseudo-out-of-sample observations) for performance evaluation. By separating the training sample from the evaluation sample, cross-validation avoids the mechanical outperformance of larger models by simulating out-of-sample model performance. Cross-validation selects models based on their predictive performance in the pseudo-out-of-sample data.

Information criteria and cross-validation each have their advantages and disadvantages. In some circumstances, AIC and cross-validation deliver asymptotically equivalent model selections (Stone, 1977; Nishii, 1984). A disadvantage of information criteria is that they are derived from certain theoretical assumptions, so if the data or models violate these assumptions, the theoretical criteria must be amended and this may be difficult or infeasible. Cross-validation implementations are often more easily adapted to account for challenging data properties like serial dependence or extreme values, and can be applied to almost any machine learning algorithm (Arlot and Celisse, 2010). On the other hand, due to its random and repetitive nature, cross-validation can produce noisier model evaluations and can be computationally expensive. Over time, the machine learning literature has migrated toward a heavy reliance on cross-validation because of its apparent adaptivity and universality, and thanks to the reduced cost in high-performance computing.

To provide a concrete perspective on model selection with cross-validation and how it fits more generally into machine learning empirical designs, we outline example designs adopted by Gu *et al.* (2020b) and a number of subsequent studies.

3.2.1 Example: Fixed Design

Let the full research sample consist of time series observations indexed $t = 1, ..., T$. We begin with an example of a fixed sample splitting scheme. The full sample of T observations is split into three disjoint subsamples. The first is the "training" sample which is used to estimate all candidate models. The training sample includes the T_{train} observations from $t = 1, ..., T_{\text{train}}$. The set of candidate models is often governed by a set of "hyperparameters" (also commonly called "tuning parameters"). An example of a hyperparameter that defines a continuum of candidate models is the shrinkage parameter in a ridge regression, while an example of a tuning parameter that describes a discrete set of models is the number of components selected from PCA.

The second, or "validation," sample consists of the pseudo-out-of-sample observations. Forecasts for data points in the validation sample are constructed based on the estimated models from the training sample. Model performance (usually defined in terms of the objective function of the model estimator) on the validation sample determines which specific hyperparameter values (i.e., which specific models) are selected. The validation sample includes the T_{validate} observations $t = T_{\text{train}} + 1, ..., T_{\text{train}} + T_{\text{validate}}$. Note that while the original model is estimated from only the first T_{train} data points, once a model specification is selected its parameters are typically re-estimated using the full $T_{\text{train}} + T_{\text{validate}}$ observations to exploit the full efficiency of the in-sample data when constructing out-of-sample forecasts.

The validation sample fits are of course not truly out-of-sample because they are used for tuning, so validation performance is itself subject to selection bias. Thus a third "testing" sample (used for neither estimation nor tuning) is used for a final evaluation of a method's predictive performance. The testing sample includes the T_{test} observations $t = T_{\text{train}} + T_{\text{validate}} + 1, ..., T_{\text{train}} + T_{\text{validate}} + T_{\text{test}}$.

Two points are worth highlighting in this simplified design example. First, the final model in this example is estimated once and for all using data through $T_{\text{train}} + T_{\text{validate}}$. But, if the model is re-estimated recursively throughout the test period, the researcher can produce more efficient out-of-sample forecasts. A reason for relying on a fixed sample split would be that the candidate models are very computationally intensive to train, so re-training them may be infeasible or incur large computing costs (see, e.g., the CNN analysis of Jiang *et al.*, 2022).

Second, the sample splits in this example respect the time series ordering of the data. The motivation for this design is to avoid inadvertent information leakage backward in time. Taking this further, it

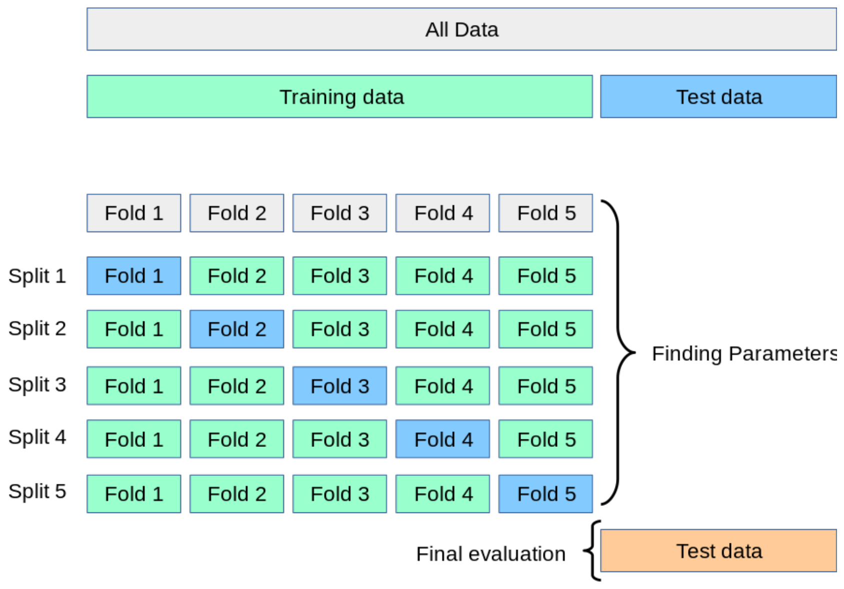

Figure 3.1: Illustration of Standard K-fold Cross-validation

Source: https://scikit-learn.org/

is common in time series applications to introduce an embargo sample between the training and validation samples so that serial correlation across samples does not bias validation. For stronger serial correlation, longer embargoes are appropriate.

Temporal ordering of the training and validation samples is not strictly necessary and may make inefficient use of data for the model selection decision. For example, a variation on the temporal ordering in this design would replace the fixed validation sample with a more traditional K-fold cross-validation scheme. In this case, the first $T_{\text{train}} + T_{\text{validate}}$ observations can be used to produce K different validation samples, from which a potentially more informed selection can be made. This would be appropriate, for example, with serially uncorrelated data. Figure 3.1 illustrates the K-fold cross-validation scheme, which creates K validation samples over which performance is averaged to make a model selection decision.

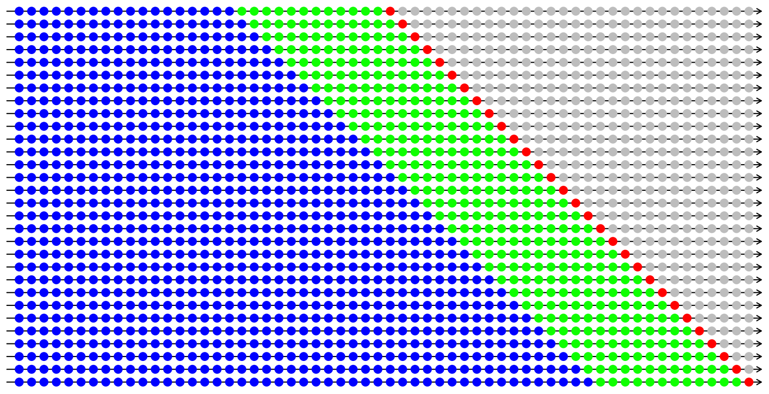

Figure 3.2: Illustration of Recursive Time-ordered Cross-validation
Note: Blue dots represent training observations, green dots represent validation observations, and red dots represent test observations. Each row represents a step in the recursive design. This illustration corresponds to the case of an expanding (rather than rolling) training window.

3.2.2 Example: Recursive Design

When constructing an out-of-sample forecast for a return realized at some time t, an analyst typically wishes to use the most up-to-date sample to estimate a model and make out-of-sample forecasts. In this case, the training and validation samples are based on observations at $1, ..., t-1$. For example, for a 50/50 split into training/validation samples, the fixed design above would be adapted to train on $1, ..., \lfloor \frac{t-1}{2} \rfloor$ and validate on $\lfloor \frac{t-1}{2} \rfloor + 1, ..., t-1$. Then the selected model is re-estimated using all data through $t-1$ and an out-of-sample forecast for t is generated.

At $t+1$, the entire training/validation/testing processes is repeated again. Training uses observations $1, ..., \lfloor \frac{t}{2} \rfloor$, validation $\lfloor \frac{t}{2} \rfloor + 1, ..., t$, and the selected model is re-estimated through t to produce out-of-sample forecast. This recursion iterates until a last out-of-sample forecast is generated for observation T. Note that, because validation is re-conducted each period, the selected model can change throughout the recursion. Figure 3.2 illustrates this recursive cross-validation scheme.

A common variation on this design is to use rolling a training window rather than an expanding window. This is beneficial if there is suspicion of structural instability in the data or if there are other modeling or testing benefits to maintaining equal training sample sizes throughout the recursion.

3.3 A Benchmark: Simple Linear Models

The foundational panel model for stock returns, against which any machine learning method should be compared, is the simple linear model. For a given set of stock-level predictive features $z_{i,t}$, the linear panel model fixes the prediction function as $g(z_{i,t}) = \beta' z_{i,t}$:

$$R_{i,t+1} = \beta' z_{i,t} + \epsilon_{i,t+1}. \tag{3.3}$$

There are a variety of estimators for this model that are appropriate under various assumptions on structure of the error covariance matrix. In empirical finance research, the most popular is Fama and Macbeth (1973) regression. Petersen (2008) analyzes the econometric properties of Fama-MacBeth and compare it with other panel estimators.

Haugen and Baker (1996) and Lewellen (2015) are precursors to the literature on machine learning for the cross section of returns. First, they employ a comparatively large number of signals: Haugen and Baker (1996) use roughly 40 continuous variables and sector dummies, and Lewellen (2015) uses 15 continuous variables. Second, they recursively train the panel linear model in (3.3) and emphasize the out-of-sample performance of their trained models. This differentiates their analysis from a common empirical program in the literature that sorts stocks into bins on the basis of one or two characteristics—which is essentially a "zero-parameter" return prediction model that asserts a functional form for $g(z_{i,t})$ without performing estimation.[4] Both Haugen and Baker (1996) and Lewellen (2015) evaluate out-of-sample model performance in the economic terms of trading strategy performance. Haugen and Baker (1996) additionally analyze international data, while Lewellen (2015) additionally analyzes model accuracy in terms of prediction R^2.

[4]To the best of our knowledge Basu (1977) is the first to perform a characteristic-based portfolio sort. He performs quintile sorts on stock-level price-earnings ratios. The adoption of this approach to by Fama and French (1992) establishes portfolio sorts as a mainstream methodology for analyzing the efficacy of a candidate stock return predictor.

Intuitively, not all information memorized by c_t needs be extracted for prediction (based on h_t).

Despite their usefulness for time series modeling, LSTM and related methods (like the gated recurrent unit of Cho *et al.*, 2014) have seen little application in the empirical finance literature. We have discussed a notable exception by Bali *et al.* (2020) above for predicting corporate bond returns. (Guijarro-Ordonez *et al.*, 2022) use RNN architectures to predict daily stock returns. Another example is Cong *et al.* (2020), who compare simple feed-forward networks to LSTM and other recurrent neural networks in monthly stock return prediction. They find that predictions from their LSTM specification outperform feed-forward counterparts, though the description of their specifications is limited and their analysis is conducted more in line with the norms of the computer science literature. Indeed, there is a fairly voluminous computer science literature using a wide variety of neural networks to predict stock returns (e.g. Sezer *et al.*, 2020). The standard empirical analysis in this literature aims to demonstrate basic proof of concept through small scale illustrative experiments and tend to focus on high frequencies (daily or intra-daily, rather than the perhaps more economically interesting frequencies of months or years) or tend to analyze one or a few assets at a time (as opposed to a more representative cross section of assets).[10] This is in contrast to the more extensive empirical analyses common in the finance and economics literature that tend to analyze monthly or annual data for large collections of assets.

3.10 Return Prediction Models For "Alternative" Data

Alternative (or more colloquially, "alt") data has become a popular topic in the asset management industry, and recent research has made strides developing machine learning models for some types of alt data. We discuss two examples in this section, text data and image data, and some supervised machine learning models customized to these alt data sources.

[10] Examples include Rather *et al.* (2015), Singh and Srivastava (2017), Chong *et al.* (2017), Bao *et al.* (2017).

3.10.1 Textual Analysis

Textual analysis is among the most exciting and fastest growing frontiers in finance and economics research. The early literature evolved from "close" manual reading by researchers (e.g. Cowles, 1933) to dictionary-based sentiment scoring methods (e.g. Tetlock, 2007). This literature is surveyed by Das *et al.* (2014), Gentzkow *et al.* (2019), and Loughran and McDonald (2020). In this section, we focus on text-based supervised learning with application to financial prediction.

Jegadeesh and Wu (2013), Ke *et al.* (2019), and Garcia *et al.* (2022) are examples of supervised learning models customized to the problem of return prediction using a term count or "bag of words" (BoW) representation of text documents. Ke *et al.* (2019) describe a joint probability model for the generation of a news article about a stock and that stock's subsequent return. An article is indexed by a single underlying "sentiment" parameter that determines the article's tilt toward good or bad news about the stock. This same parameter predicts the direction of the stock's future return. From a training sample of news articles and associated returns, Ke *et al.* (2019) estimate the set of most highly sentiment-charged (i.e., most return-predictive) terms and their associated sentiment values (i.e., their predictive coefficients). In essence, they provide a data-driven methodology for constructing sentiment dictionaries that are customized to specific supervised learning tasks.

Their method, called "SESTM" (Sentiment Extraction via Screening and Topic Modeling), has three central components. The first step isolates the most relevant terms from a very large vocabulary of terms via predictive correlation screening. The second step assigns term-specific sentiment weights using a supervised topic model. The third step uses the estimated topic model to assign article-level sentiment scores via penalized maximum likelihood.

SESTM is easy and cheap to compute. The model itself is analytically tractable and the estimator boils down to two core equations. Their modeling approach emphasizes simplicity and interpretability. Thus, it is "white box" and easy to inspect and interpret, in contrast to many state-of-the-art NLP models built around powerful yet opaque neural

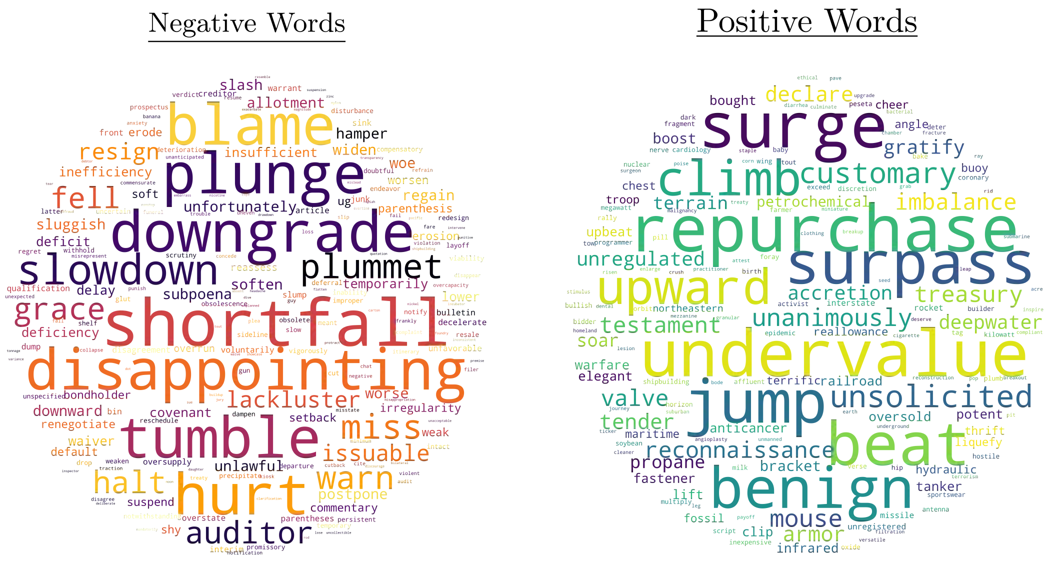

Figure 3.8: Sentiment-charged Words

Note: This figure reports the list of words in the sentiment-charged set S. Font size of a word is proportional to its average sentiment tone over all 17 training samples.

network embedding specifications. As an illustration, Figure 3.8 reports coefficient estimates on the tokens that are the strongest predictors of returns in the form of a word cloud. The cloud is split into tokens with positive or negative coefficients, and the size of each token is proportional to the magnitude of its estimated predictive coefficient.

Ke *et al.* (2019) devise a series of trading strategies to demonstrate the potent return predictive power of SESTM. In head-to-head comparisons, SESTM significantly outperforms RavenPack (a leading commercial sentiment scoring vendor used by large asset managers) and dictionary-based methods such as Loughran and McDonald (2011).

A number of papers apply supervised learning models to BoW text data to predict other financial outcomes. Manela and Moreira (2017) use support vector regression to predict market volatility. Davis *et al.* (2020) use 10-K risk factor disclosure to understand firms' differential return responses to the COVID-19 pandemic, leveraging Taddy (2013)'s multinomial inverse regression methodology. Kelly *et al.* (2018) introduce a method called hurdle distributed multinomial regression (HDMR) to improve count model specifications and use it to build a text-based index measuring health of the financial intermediary sector.

These analyses proceed in two general steps. Step 1 decides on the numerical representation of the text data. Step 2 uses the representations as data in an econometric model to describe some economic phenomenon (e.g., asset returns, volatility, and macroeconomic fundamentals in the references above).

The financial text representations referenced above have some limitations. First, all of these examples begin from a BoW representation, which is overly simplistic and only accesses the information in text that is conveyable by term usage frequency. It sacrifices nearly all information that is conveyed through word ordering or contextual relationships between terms. Second, the ultra-high dimensionality of BoW representations leads to statistical inefficiencies—Step 2 econometric models must include many parameters to process all these terms despite many of the terms conveying negligible information. Dimension reductions like LDA and correlation screening are beneficial because they mitigate the inefficiency of BoW. However, they are derived from BoW and thus do not avoid the information loss from relying on term counts in the first place. Third, and more subtly, the dimension-reduced representations are *corpus specific*. For example, when Bybee *et al.* (2020) build their topic model, the topics are estimated only from *The Wall Street Journal*, despite the fact that many topics are general language structures and may be better inferred by using additional text outside of their sample.

Jiang *et al.* (2023) move the literature a step further by constructing refined news text representations derived from so-called "large language models" (LLMs). They then use these representations to improve models of expected stock returns. LLMs are trained on large text data sets that span many sources and themes. This training is conducted by specialized research teams that perform the Herculean feat of estimating a general purpose language model with astronomical parameterization on truly big text data. LLM's have billions of parameters (or more) and are trained on billions of text examples (including huge corpora of complete books and massive portions of the internet). But for each LLM, this estimation feat is performed once, then the estimated model is made available for distribution to be deployed by non-specialized researchers in downstream tasks.

In other words, the LLM delegates Step 1 of the procedure above to the handful of experts in the world that can best execute it. A Step 2 econometric model can then be built around LLM output. Like LDA (or even BoW), the output of a foundation model is a numerical vector representation (or "embedding") of a document. A non-specialized researcher obtains this output by feeding the document of interest through software (which is open-source in many cases). The main benefit of an LLMS in Step 1 is that it provides more sophisticated and well-trained text representations than used in the literature referenced above. This benefit comes from the expressivity of heavy nonlinear model parameterizations and from training on extensive language examples across many domains, throughout human history, and in a wide variety of languages. The transferability of LLMs make this unprecedented scale of knowledge available for finance research.

Jiang *et al.* (2023) analyze return predictions based on a news text processed through a number of LLMs including Bidirectional Encoder Representations from Transformer (BERT) of Devlin *et al.* (2018), Generative Pre-trained Transformers (GPT) of Radford *et al.* (2019), and Open Pre-trained Transformers (OPT) by Zhang *et al.* (2022). They find that predictions from pre-trained LLM embeddings outperform prevailing text-based machine learning return predictions in terms of out-of-sample trading strategy performance, and that the superior performance of LLMs stems from the fact that they can more successfully capture contextual meaning in documents.

3.10.2 Image Analysis

Much of modern machine learning has evolved around the task of image analysis and computer vision, with large gains in image-related tasks deriving from the development of convolutional neural network (CNN) models. Jiang *et al.* (2022) introduce CNN image analysis techniques to the return prediction problem.

A large finance literature investigates how past price patterns forecast future returns. The philosophical perspective underpinning these analyses is most commonly that of a hypothesis test. The researcher formulates a model of return prediction based on price trends—such as a regression of one-month-ahead returns on the average return over the previous twelve months—as a test of the weak-form efficient markets null hypothesis. Yet it is difficult to see in the literature a specific alternative hypothesis. Said differently, the price-based return predictors studied in the literature are by and large ad hoc and discovered through human-intensive statistical learning that has taken place behind the curtain of the academic research process. Jiang *et al.* (2022) reconsider the idea of price-based return predictability from a different philosophical perspective founded on machine learning. Given recent strides in understanding how human behavior influences price patterns (e.g. Barberis and Thaler, 2003; Barberis, 2018), it is reasonable to expect that prices contain subtle and complex patterns about which it may be difficult to develop specific testable hypotheses. Jiang *et al.* (2022) devise a systematic machine learning approach to elicit return predictive patterns that underly price data, rather than testing specific ad hoc hypotheses.

The challenge for such an exploration is balancing flexible models that can detect potentially subtle patterns against the desire to maintain tractability and interpretability of those models. To navigate this balance, Jiang *et al.* (2022) represent historical prices as an image and use well-developed CNN machinery for image analysis to search for predictive patterns. Their images include daily opening, high, low, and closing prices (ofter referred to as an "OHLC" chart) overlaid with a multi-day moving average of closing prices and a bar chart for daily trading volume (see Figure 3.9 for a related example from Yahoo Finance).

A CNN is designed to automatically extract features from images that are predictive for the supervising labels (which are future realized returns in the case of Jiang *et al.*, 2022). The raw data consists of pixel value arrays. A CNN typically has a few core building blocks that convert the pixel data into predictive features. The building blocks are stacked together in various telescoping configurations depending on the application at hand. They spatially smooth image contents to reduce noise and accentuate shape contours to maximize correlation of images with their labels. Parameters of the building blocks are learned as part

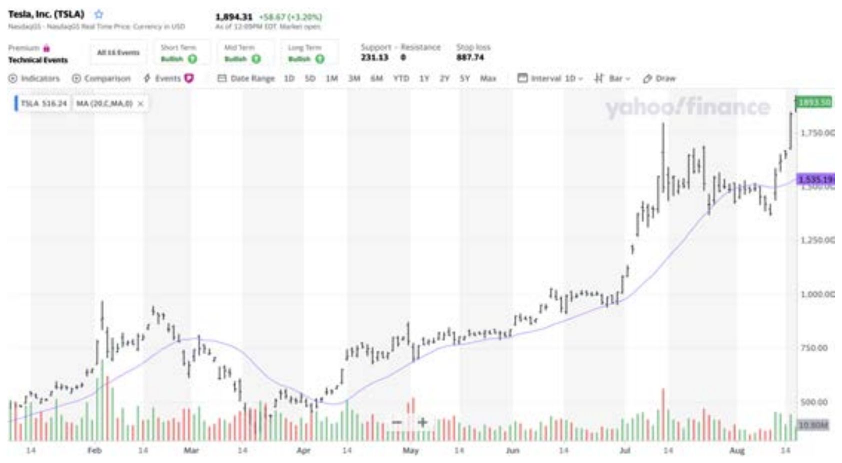

Figure 3.9: Tesla OHLC Chart from Yahoo! Finance

Note: OHLC chart for Tesla stock with 20-day moving average price line and daily volume bars. Daily data from January 1, 2020 to August 18, 2020.

of the model estimation process.

Each building block consists of three operations: convolution, activation, and pooling. “Convolution” is a spatial analogue of kernel smoothing for time series. Convolution scans through the image and, for each element in the image matrix, produces a summary of image contents in the immediately surrounding area. The convolution operates through a set of learned “filters,” which are low dimension kernel weighting matrices that average nearby matrix elements.

The second operation in a building block, “activation,” is a nonlinear transformation applied element-wise to the output of a convolution filter. For example, a “Leaky ReLU” activation uses a convex piecewise linear function, which can be thought of as sharpening the resolution of certain convolution filter output.

The final operation in a building block is “max-pooling.” This operation uses a small filter that scans over the input matrix and returns the maximum value the elements entering the filter at each location in the image. Max-pooling acts as both a dimension reduction device and as a de-noising tool.

Figure 3.10 illustrates how convolution, activation, and max-pooling combine to form a basic building block for a CNN model. By stacking many of these blocks together, the network first creates representations

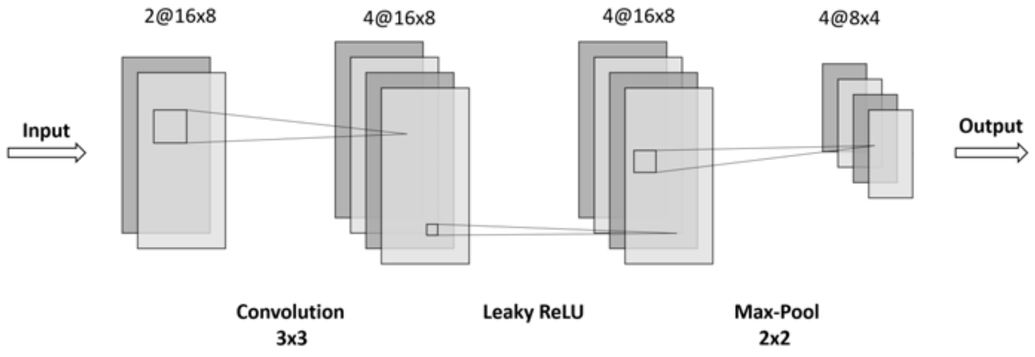

Figure 3.10: Diagram of a Building Block

Note: A building block of the CNN model consists of a convolutional layer with 3×3 filter, a leaky ReLU layer, and a 2×2 max-pooling layer. In this toy example, the input has size 16×8 with 2 channels. To double the depth of the input, 4 filters are applied, which generates the output with 4 channels. The max-pooling layer shrinks the first two dimensions (height and width) of the input by half and keeps the same depth. Leaky ReLU keeps the same size of the previous input. In general, with input of size $h \times w \times d$, the output has size $h/2 \times w/2 \times 2d$. One exception is the first building block of each CNN model that takes the grey-scale image as input: the input has depth 1 and the number of CNN filters is 32, boosting the depth of the output to 32.

of small components of the image then gradually assembles them into representations of larger areas. The output from the last building block is flattened into a vector and each element is treated as a feature in a standard, fully connected feed-forward layer for the final prediction step.[11]

Jiang *et al.* (2022) train a panel CNN model to predict the direction of future stock returns using daily US stock data from 1993 to 2019. A weekly rebalanced long-short decile spread trading strategy based on the CNN-based probability of a positive price change earns an out-of-sample annualized Sharpe ratio of 7.2 on an equal-weighted basis and 1.7 on a value-weighted basis. This outperforms well known price trend strategies—various forms of momentum and reversal—by a large and significant margin. They show that the image representation is a key driver of the model’s success, as other time series neural network specifications have difficulty matching the image CNN’s performance. Jiang

[11]Ch. 9 of Goodfellow *et al.* (2016) provide a more general introduction to CNN.

et al. (2022) note that their strategy may be partially approximated by replacing the CNN forecast with a simpler signal. In particular, a stock whose latest close price is near the bottom of its high-low range in recent days tends to appreciate in the subsequent week. This pattern, which is not previously studied in the literature, is detected from image data by the CNN and is stable over time and across market size segments and across countries.

Glaeser *et al.* (2018) study the housing market using images of residential real estate properties. They use a pre-trained CNN model (Resnet-101 from He *et al.*, 2016) to convert images into a feature vector for each property, which they then condense further using principal components, and finally the components are added to an otherwise standard hedonic house pricing model. Glaeser *et al.* (2018) find that property data derived from images improves out-of-sample fit of the hedonic model. Aubry *et al.* (2022) predict art auction prices via a neural network using artwork images accompanied by non-visual artwork characteristics, and use their model to document a number of informational inefficiencies in the art market. Obaid and Pukthuanthong (2022) apply CNN to classify photos on the *Wall Street Journal* and construct a daily investor sentiment index, which predicts market return reversals and trading volume. The association is strongest among stocks with more severe limits-to-arbitrage and during periods of elevated risk. They also find that photos convey alternative information to news text.[12]

[12]Relatedly, Deng *et al.* (2022) propose a theoretical model to rationalize how the graphical content of 10-K reports affects stock returns.