The Best of Both Worlds? Predicting Realized Volatility Based on Convolutional Neural Nets and Intraday Return Images

Niklas Betz* Thomas L.A. Heil† Franziska J.Peter ‡

Abstract

We predict daily realized volatility based on intraday stock return images which are fed into convolutional neural networks. We combine these predictors within a linear model mimicking the cascading structure of the Heterogeneous Autoregressive Model (HAR) model. Predicting daily realized volatility with this model for a sample of Dow Jones constituent stocks shows that the trained convolutional neural networks can extract predictive patterns from the images of return series. These patterns provide information beyond those of the traditional HAR components. Our findings highlight the potential for improved forecasting performance when relying on deep learning approaches.

Keywords: realized volatility forecasting, HAR model, convolutional neural nets

*Zeppelin University; n.betz@zeppelin-university.net

†Zeppelin University; Thomas.heil@zu.de

‡Corresponding author; Zeppelin University, Am Seemooser Horn 20, 88046, Friedrichshafen; Germany; franziska.peter@zu.de;

1 Introduction

Financial volatility is an important input for security valuation, portfolio allocation, investment decisions, risk management, and monetary policy. Accurate volatility forecasting models have been extensively studied by both academics and practitioners (e.g. Poon and Granger, 2003, Hansen and Lunde, 2011, Hong et al., 2021, Blair et al., 2001). The Heterogeneous Autoregressive (HAR) model introduced by Corsi (2009) serves as a primary benchmark for volatility prediction in the current finance literature. The HAR model relies on realized volatility (RV) calculated from high-frequency data (Andersen and Teräsvirta, 2009 and Barndorff-Nielsen and Shephard, 2002) and effectively captures the persistence of daily volatility in a concise linear way.

More recently, using deep neural networks (NNs) to forecast volatility has gained traction, challenging the dominance of traditional models. NNs do not rely on restrictive assumptions regarding data-generating processes and can detect complex non-linear patterns (Kristjanpoller and Minutolo, 2015). Specifically, Convolutional Neural Networks (CNNs) have gained popularity in financial forecasting, consistently outperforming traditional price trend indicators and benchmark volatility models (Audrino and Faessler, 2023; Borovykh et al., 2018; Bucci, 2020; Jiang et al., 2023; Miura et al., 2019; Rahimikia and Poon, 2020; Reisenhofer et al., 2022; Zhang et al., 2023; Zhu et al., 2023).

CNNs were originally developed for image analysis tasks (LeCun et al., 1998) and excel at deriving patterns from 2D spatial representation. Consequently, rather than defining an underlying stochastic process, CNNs possess the capability to extract patterns directly from an image of a return series. Jiang et al. (2023) and Sezer and Ozbayoglu (2018) utilize CNNs to predict stock returns and develop trading strategies, demonstrating superior performance compared to selected benchmarks. Most recently, Reisenhofer et al. (2022) introduced HARNet, a neural network (NN) that utilizes hierarchies of dilated convolutional layers to compute features resembling the weekly and monthly aggregates of the original HAR model. Audrino and Faessler (2023) use CNNs as classifiers for price patterns which they subsequently use to augment the HAR model. Their analysis also shows improvements in forecasting performance, when including CNN based predictors within the traditional HAR approach.

In contrast to previous studies we use three deep CNNs to directly predict daily RV based on intraday return images, which mirror the daily, weekly, and monthly volatility components. The resulting predictors are subsequently combined in one common forecast. Thereby, our approach is theoretically based on the Heterogeneous Market Hypothesis (Müller et al., 1997, Corsi, 2009), where groups of investors make decisions based on different investment horizons and daily RV emerges as a combination of these short, medium and long-term traders' actions.

Within the traditional HAR approach daily volatility emerges based on an AR(1) structure of these cascading volatility components. Our Convolutional Neural Nets based approach can be viewed as a generalization of this structure. Since each CNN is fed images of the past day's, week's, and month's intraday return series, respectively, the RV predictions are based on a richer information set, because CNNs are able to use higher frequency (intraday) data to directly predict a daily target variable (RV). Furthermore, these CNN-RV predictors are based on more complex, non-linear relationships between past return images and the next day's daily RV. Thereby, our approach seeks to use the advantage of deep learning methods such as the CNN in terms of flexibility and overcome - to some extent - their notion of a black-box via their combination in a HAR-like cascading fashion to preserve the interpretability against the background of the Heterogeneous Market Hypothesis.

In detail, we train three CNNs based on past day's, week's and month's minute-by-minute return images for a sample of 28 Jones Industrial Average Index (DJIA) intraday stock returns. Combining the resulting daily, weekly, and monthly predictors, we forecast daily RV for each stock. We use data from 2010 through 2019 to train and test the model. We evaluate the models by testing the in-sample statistical significance of the CNN components and by assessing the out-of-sample forecasting performance of the CNN model against the benchmark of the traditional

HAR model. We find that the CNN model outperforms the baseline HAR model in forecasting daily realized volatility. Including the CNN predictors as additional regressors in the benchmark HAR model reveals that mostly both groups of predictors significantly contribute to RV prediction, while the overall performance of the combined model does not exceed the purely CNN predictor based approach.

The remainder of this paper is structured as follows: In Section 2.1, we outline the general structure of the benchmark HAR, the CNNs-based and the combined models. Section 2.2 develops the images of intraday stock returns used as inputs for the CNNs, section 2.3 outlines the basics of CNNs, and section 2.4 describes the architecture of the CNNs applied in this study. Section 3 provides information on the data. Sections 4.1 and 4.2 present the in-sample and out-of-sample results, respectively. Finally, Section 5 concludes the paper.

2 Forecasting Realized Volatility from Intraday Return Images

2.1 The General Structure of the forecasting models

The realized volatility forecasting approach proposed in this paper maintains the parsimonious structure of the standard HAR model based on daily, weekly, and

monthly RV components (for details on the derivation of the HAR model, we refer to Appendix A and Corsi, 2009). The benchmark HAR model is given by

$$RV_{i,t+1d}^{(d)} = c + \beta_i^{(d)} RV_{i,t}^{(d)} + \beta_i^{(w)} RV_{i,t}^{(w)} + \beta_i^{(m)} RV_{i,t}^{(m)} + \omega_{i,t+1d} \tag{1}$$

where $RV_{i,t}^{(d)}$, $RV_{i,t}^{(w)}$, and $RV_{i,t}^{(m)}$ denote the daily, weekly, and monthly realized volatility aggregates, as described in Corsi (2009) for stock i at day t, which are derived from the previous day's, the past 5 days', and the past 22 days' RV estimators, respectively. $\omega_{i,t+1d}$ denotes contemporaneously and serially independent zero-mean innovations.

Our proposed CNN-RV model relies on three CNN-based prediction components, aligning with the standard daily, weekly, and monthly components. This approach aims to capture more complex abstractions by directly relating past days' intraday returns to the next day's daily RV, potentially enhancing the overall model performance. We preserve the general structure of the HAR model, which allows us to analyze the importance of different components within the prediction task. Our proposed CNN-RV model has the following structure:

$$RV_{i,t+1d}^{(d)} = a + \gamma_i^{(d)} \mathrm{CNN}^{(d)} \left[I_{i,t}^{(d)} \right] + \gamma_i^{(w)} \mathrm{CNN}^{(w)} \left[I_{i,t}^{(w)} \right] + \gamma_i^{(m)} \mathrm{CNN}^{(m)} \left[I_{i,t}^{(m)} \right] + \epsilon_{i,t+1d} \tag{2}$$

where $I_{i,t}^{(d)}$ is an image of stock $i's$ minute-by-minute intraday returns at date t. Analogously, $I_{i,t}^{(w)}$ denotes the image of the intraday returns starting at date $[t-4]$ up until date t of stock i, i.e., for a total of five days. Lastly, $I_{i,t}^{(m)}$ denotes the image of stock i's intraday returns from $[t-21]$ until t, for a total of 22 days. $\text{CNN}^{(d)}$, $\text{CNN}^{(w)}$, and $\text{CNN}^{(m)}$ refer to separate CNNs which predict day-ahead RV based on the daily, weekly, and monthly input images, respectively. Each of these CNNs is trained exclusively on the images $(I_{i,t})$ at the time range indicated by the CNNs' superscript. The CNNs lack stock-specific (i) subscripts, as one common CNN is trained on the images of all stocks at the respective aggregation period.[1] Then, for example, $\text{CNN}^{(w)}\left[I_{i,t}^{(w)}\right]$ denotes the prediction for $RV_{i,t+1d}^{(d)}$ produced by the trained $\text{CNN}^{(w)}$ when fed the images $I_{i,t}^{(w)}$ as input. The subscripts i on the coefficients indicate that a separate CNN-RV model is fitted for each stock i. $\epsilon_{i,t+1d}$ denotes contemporaneously and serially independent zero-mean innovations. A constant term a is included when fitting the regression models.

Finally, a combination of the standard and CNN-based components results in what we refer to as the CNN-HAR model:

$$\begin{aligned} RV_{i,t+1d}^{(d)} = d &+ \tilde{\beta}_i^{(d)} RV_{i,t}^{(d)} + \tilde{\beta}_i^{(w)} RV_{i,t}^{(w)} + \tilde{\beta}_i^{(m)} RV_{i,t}^{(m)} \\ &+ \tilde{\gamma}_i^{(d)} \text{CNN}^{(d)}\left[I_{i,t}^{(d)}\right] + \tilde{\gamma}_i^{(w)} \text{CNN}^{(w)}\left[I_{i,t}^{(w)}\right] + \tilde{\gamma}_i^{(m)} \text{CNN}^{(m)}\left[I_{i,t}^{(m)}\right] \\ &+ \varepsilon_{i,t+1d} \end{aligned} \tag{3}$$

[1] Training one common CNN for all stocks at each period increases training data availability.

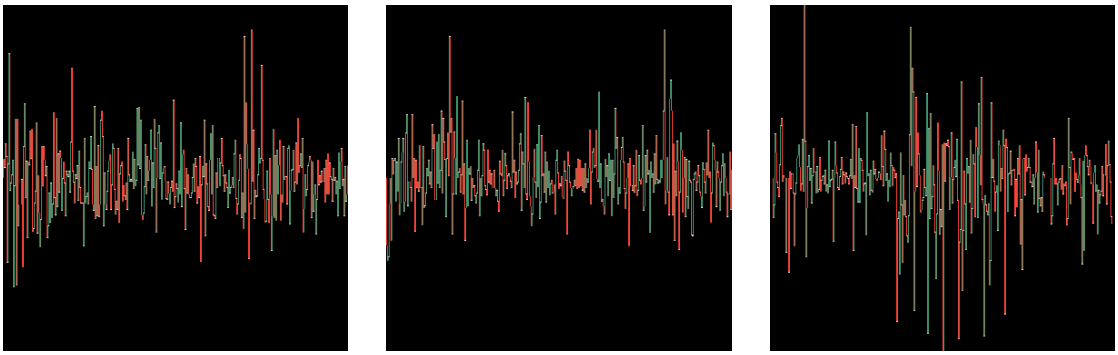
The CNN-HAR model allows us to assess whether a combination of both approaches yields a superior forecasting performance. All models are fit by ordinary least squares and relying on Newey-West standard errors.

2.2 Time Series to Image Encoding

Financial time series data are typically one-dimensional (1D), while CNNs were originally designed for two-dimensional (2D) spatial objects, such as images (Goodfellow et al., 2016; Wang et al., 2020). Encoding the input data, denoted by $I_{i,t}^{(.)}$ for CNNs is therefore a crucial step in this process. Mapping a stream of 1D financial time series data into the required matrix form has not yet reached a consensus in the literature. We therefore have adopted an image design similar to the approach used by Jiang et al. (2023) and Sezer and Ozbayoglu (2019). However, we employ a colored waterfall-chart-type graph. In this representation, increasing returns are represented by green-colored vertical bars, while decreasing returns are represented by red-colored bars. Each pixel within the image is represented by a triplet of red, green, and blue (R, G, B) values. The black image background corresponds to pixel vectors of (0, 0, 0). As each pixel triplet value constitutes a different CNN input channel, we allow for potentially different treatments of positive and negative returns within the CNN, when predicting daily realized volatility. The intraday log returns are concatenated according to time along the x-axis. Relying on a

minute-by-minute sampling frequency, the daily images $I_{i,t}^{(d)}$ are 380 pixels wide. Missing values are represented as black pixels. While the x-axis represents time, the y-axis refers to the intraday log returns. We define square images of 380 pixels in each dimension and determine the scale of the y-axis in the daily graphs to be ± 0.0025, which includes 99% of the observations. All returns beyond this range are cut off at the limiting value. In effect, they are treated as outliers. Using a fixed y-axis scale for all images allows the CNN to effectively distinguish between high- and low-volatility patterns, as volatility depends on the absolute variation of intraday returns.

Figure 1: Image Design
The graph shows three exemplary images corresponding to the daily, weekly, and monthly time range.



$I^{(d)}_{\text{Boeing, 2019/02/20}}$ $I^{(w)}_{\text{IBM, 2017/04/11}}$ $I^{(m)}_{\text{Visa, 2013/12/31}}$

The most literal translation of this design for the weekly and monthly components of $I_{i,t}^{(w)}$ and $I_{i,t}^{(m)}$ in equation (2) would be to increase the width of the images to include the minutely intraday returns of the entire aggregation periods. This

design would yield very large images, which increases computational cost drastically. However, reducing the sampling frequency of the returns should equally capture relevant information for longer-term traders, since their actions are less dependent on very high-frequency returns. For this reason, we define all three image types, i.e., $I_{i,t}^{(d)}$, $I_{i,t}^{(w)}$, and $I_{i,t}^{(m)}$ to have the same dimensions by reducing the sampling frequency of the depicted weekly and monthly return images accordingly: Concatenating five days of five-minute-by-minute returns yields an image of 380×380 pixels, where the first pixel column represents the first return of day $[t-4]$, while the ultimate pixel column on the right presents the last five minutely return of day t. For the monthly images, we reduced the sampling frequency to 22 minutes, which yields an image width of 374 pixels. We add three blank columns to the left and right of this return series to maintain a total width of 380 pixels. As the returns for the weekly and monthly images are calculated at lower frequencies, the absolute range of the y-axis must increase correspondingly. We determine a scale of ± 0.005 for the weekly and a scale of ± 0.01 for the monthly images. Again, this includes more than 99% of all intraday observations in the images. Figure 1 depicts exemplary images representing the daily, weekly, and monthly return series. These are fed into three separate CNNs, which then produce the components $\text{CNN}^{(d)}[I_{i,t}^{(d)}]$, $\text{CNN}^{(w)}[I_{i,t}^{(w)}]$, and $\text{CNN}^{(m)}[I_{i,t}^{(m)}]$ for the CNN-RV model in equation 2.

2.3 Convolutional Neural Nets

CNNs are a group of regularized, feed-forward neural networks that learn feature engineering via filter optimization. More specifically, each layer of a CNN consists of a convolution operation, a (non-linear) activation, and a pooling operation to extract possibly non-linear features from the image. An input layer and an output layer wrap these specific CNN layers. Additionally, a fully connected layer is applied between the last CNN layer and the output layer to map the extracted features to a single output. The basic architecture of our CNN model is similar to the one employed in Jiang et al. (2023). We employ three sequential building blocks consisting of a convolutional and max-pooling layer each. These three building blocks are followed by a fully connected layer, with the final output layer comprising a single unit. CNNs are built by sequentially applying convolution, activation, and pooling, creating representations of small details combined in deeper layers to capture more complex features in larger areas of the original image. The final building block is flattened and passed to a fully connected layer, followed by an output layer for prediction. These last layers of the CNN are analogous to a regular multilayer perceptron, where each unit is connected to all units in the previous layer. They thereby produce a prediction via a combination of the learned image features.[2]

[2]For a more in-depth introduction to CNNs see Goodfellow et al. (2016) and Li et al. (2022).

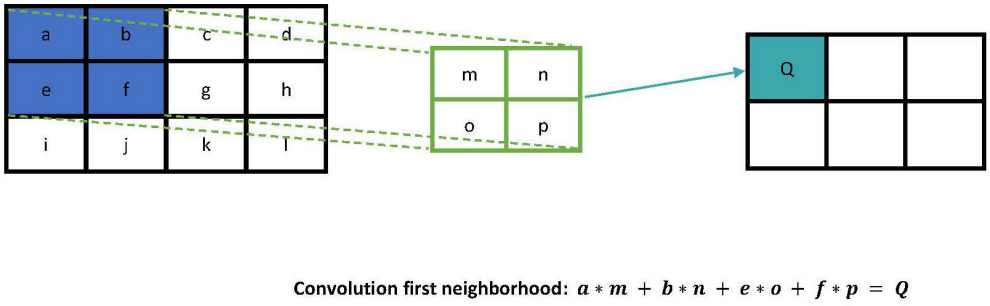
Input Layer

The input layer provides the data suitable for the CNN calculations to the network. Therefore, the input layer provides the CNN with the images as a 2-D grid of pixels. The CNN processes one image at a time unlike a classic feedforward neural network in which the input layer simultaneously provides a set of input features to the network.

Convolution Operation

Via the convolution operation, filters are applied to small subsets (areas) of an input image, producing activation maps that represent the responses of the filters at different positions. By learning these filters in training, CNNs can detect features with high predictive power, regardless of their position in the image. The functionality of these filters is adjusted by three hyperparameters: the stride with which the filter is slid across the input (image), the number of filters, and the (two-dimensional) filters' size. The output of each convolution layer consists of as many depthwise stacked activation maps as filters used within this layer. A non-linear activation is applied to the convolution output before passing it to the next layer. Figure 2 illustrates the first step of the convolutional operation. The 2×2 filter (green matrix) will continue to slide over the input matrix according to the defined or hypertuned stride length to generate the output matrix.

Figure 2: Illustration of the Convolution Operation



Activation

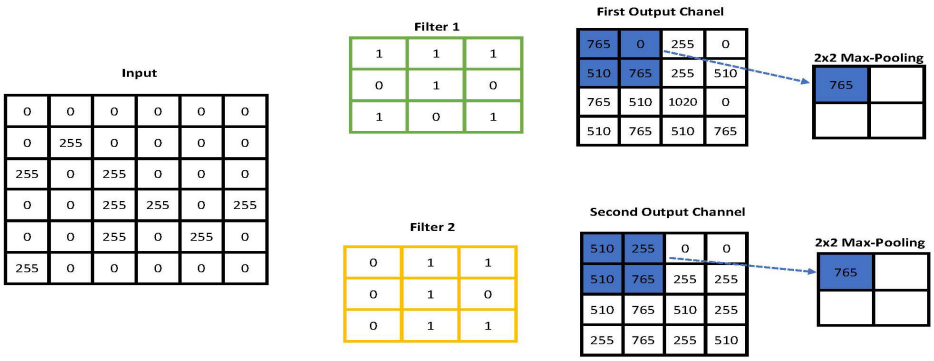
For different inputs, different neurons are essential to make a prediction. More specifically, with different inputs, some neurons "fire" and some are "silent". The activation function provides a way to provide weights and biases to each neuron. Moreover, it introduces non-linearity to the neural network's input as neural networks without an activation function collapse to a linear regression. The introduction of non-linearity allows the neural network to find more complex patterns in the mapping from the input to the output. In the case of CNNs, the activation function is applied to the output of the convolution operations and the fully connected layers.

Pooling Operation

Pooling layers are commonly used after one or multiple convolution layers. They downsample the image size and control overfitting by de-noising the input. In

contrast to convolution, pooling applies a fixed function to each activation map, reducing the number of parameters. The pooling function typically takes the form of max-pooling, which outputs the maximum value of the entries the filter covers at each position. Figure 3 illustrates the first step of the max pooling operation. Using two filters and convolutions, two output matrices are generated, which are then reduced to two 2×2 matrices by subsequently applying the max pooling operation to each 2×2 segment of the output channels.

Figure 3: Illustration of the Max-Pooling Operation



Flattening and Fully Connected Layer

The convolution and pooling layer produces a compromised but still one-dimensional output. In order to generate a prediction using the extracted features, a reshaping in a one-dimensional structure is applied. The two-dimensional grid of extracted features is flattened to represent a vector of explanatory variables. The flattened vector is an input layer for a fully connected dense layer similar to those used in

standard feed-forward neural networks. The fully connected layer processes the filtered features from the images as inputs similar to a regression problem and, ultimately, provides an automated selection of predictors for the output layer.

Output Layer

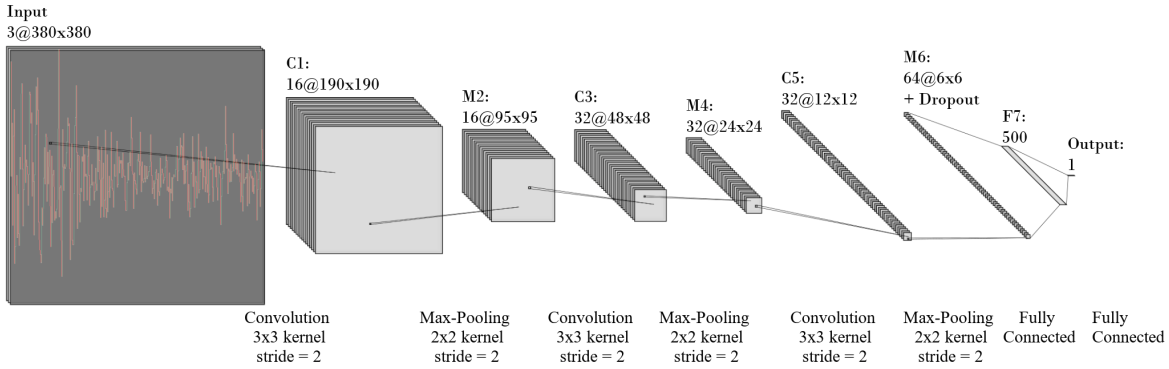
The final layer in an artificial neural network, the output layer, produces the prediction based on the feature extraction beforehand. The output layer has weights and biases and is similar to a non-linear regression equation, predicting the next day's RV. Therefore, based on the predicted value, a loss function is calculated, which in return provides a measure of the performance of the current set of weights in the CNN. Hence, the loss is used to re-adjust the weights in the back-propagation step of neural networks. By re-applying the forward- and backward-propagation, the neural network learns to adjust the internal weights to increase the predictive performance.

2.4 CNN architecture for realized volatility prediction

The inputs into the CNNs are the images discussed in Section 2.2. With their dimension of 380x380x3, the images are much larger than is common in the related literature. This large size results in highly parameterized CNNs, introducing large computational needs. Therefore, we increase the stride of the convolution operation to two. This results in a halving of the image width and height in

each convolutional layer, thus decreasing the computational burden for the entire CNN (Goodfellow et al., 2016). We apply max-pooling after each convolutional layer to reduce noise. Figure 4 depicts the architecture of the CNNs we train. Grid-search-based hyperparameter tuning yields an optimal initial learning rate of

Figure 4: Final CNN Model



10^{-4} for the adaptive Adam optimizer (Kingma & Ba, 2014) and a batch size of 8. Further, the optimal number of filters in the convolutional layers is determined to be 16 in C1, 32 in C3, and 64 in C5 (see Figure 4). Lastly, tuning the kernel size results in the adoption of a 3x3 kernel for all three convolutional layers. Our CNNs apply the Rectified Linear Unit (ReLU) activation function to the convolution outputs. (Goodfellow et al., 2016).[3] We introduce a dropout layer for regularization between the final pooling operation and the fully connected layer (Srivastava et al., 2014). Furthermore, the dropout rate and number of units in the fully connected

[3]The ReLU activation function takes the following form: $\text{ReLU(x)} = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise.} \end{cases}$

layer for each CNN are jointly tuned. The results are presented in Table A.1 in the Appendix. A dropout rate of 0.75 with 500 nodes in the dense layer is within the ten best hyperparameter combinations for all three CNNs and is revealed as the best choice for $\text{CNN}^{(m)}$ and as the second best for $\text{CNN}^{(d)}$. This finding suggests that similar hyperparameter choices for all three CNNs are appropriate. Therefore, we adopt this combination for all three CNNs, producing three identical CNN models that differ only in the data they training data.

To produce reliable results in the face of CNN training stochasticity (Gu et al., 2020; Jiang et al., 2023), we train an ensemble of four CNNs at each frequency and average their forecasts. Table A.2 in the Appendix depicts the evolution of average validation root mean squared errors (RMSE) in these ensembles until training epoch 20. For all three CNNs, the average RMSE at the fourteenth epoch is within the lowest five. Therefore, for simplicity, we train each CNN for fourteen epochs. CNNs are trained to minimize the MSE of day-ahead RV predictions. The CNNs are trained exclusively on their respective training sets and evaluated on the validation set.

3 Data

We use high-frequency return data from 28 stocks, which have been constituents of DJIA during our entire sample period. The sample period runs from January

4th, 2010, to December 31st, 2019, which yields 2516 trading days per stock. Each day covers the trading window from 9:36 a.m. to 3:55 p.m. ET, with log returns sampled minute-by-minute. Therefore, one day usually consists of 380 intraday observations at a minute-by-minute frequency. Daily RV is calculated based on squared one-minute returns. Given the high liquidity of the DJIA constituents' stocks, this seems an appropriate frequency for calculating RV (Liu et al., 2015). We split the data into a training-, validation-, and test set along the temporal dimension (Golbayani et al., 2020). We perform a 70-15-15 split, such that the first 1762 days of each stock, from January 4th, 2010, until December 30th, 2016, are allocated to the training set. The validation set comprises each stock's 376 trading days from January 3rd, 2017, until June 29th, 2018. Lastly, the test set starts on July 2nd, 2017, and extends 377 days until December 30th, 2019. Following Rahimikia and Poon (2020), we use the RMSE to evaluate forecasting performance (Patton, 2011).

4 Realized Volatility Prediction

4.1 In-Sample Results

Table 1 shows in-sample results on the benchmark HAR and the CNN-RV model according to equations 1 and 2, respectively. We observe a higher R^2 for the CNN-RV model compared to the benchmark HAR model for all stocks. However, it has to be noted that the CNN components are produced by a highly complex, non-linear function approximator. The Universal Approximation Theorem states that Neural Networks can approximate any possible continuous function with reasonable accuracy (Cybenko, 1989). Continued in-sample fitting of our CNNs for further epochs would have decreased training loss as the CNN starts overfitting. However, the CNN predictors are optimized for accurate out-of-sample forecasting rather than explaining variations within the training sample. Interestingly, our results indicate that the CNN components are already highly significant at the point at which we determined the training to stop. Table 2 depicts the estimated parameters for the combined model according to equation 3.[4] For 29 out of 84 coefficient pairs we observe that the benchmark and the CNN components are significant at the same aggregation period. This finding suggests that both inputs contribute to explaining next day's RV variations.

[4]We observe only partly significant coefficients with mostly negative signs considering the benchmark HAR components. The latter is not surprising considering the substantial correlation between the different components (see Table A.3 in the Appendix).

Table 1: In-Sample Results: HAR and CNN-RV Models.

The table displays the coefficient estimates of fitting the HAR model (equation 1) and the CNN-RV model (equation 2). Robust standard errors are given in parentheses. ***, **, * indicate significance at the 1%, 5%, and 10% significance level respectively. F_{HAR} and F_{CNN-RV} columns give the p-value of the F-test for joint significance of the variables referred to in the subscript.

Stock	HAR				CNN-RV				F-test	
	$\beta_i^{(d)}$	$\beta_i^{(w)}$	$\beta_i^{(m)}$	R^2_{HAR}	$\gamma_i^{(d)}$	$\gamma_i^{(w)}$	$\gamma_i^{(m)}$	R^2_{CNN}	F_{HAR}	F_{CNN-RV}
American Express	0.497***	0.195***	0.226***	0.619	0.457***	0.354***	0.284***	0.737	0.000	0.000
	[0.050]	[0.057]	[0.049]		[0.051]	[0.084]	[0.077]			
Amgen	0.416***	0.338***	0.149***	0.574	0.476***	0.267***	0.420***	0.659	0.034	0.000
	[0.043]	[0.054]	[0.042]		[0.055]	[0.072]	[0.065]			
Apple	0.495***	0.094*	0.218***	0.386	0.784***	0.334	−0.020	0.604	0.067	0.000
	[0.061]	[0.056]	[0.053]		[0.155]	[0.247]	[0.366]			
Boeing	0.425***	0.227***	0.229***	0.487	0.399***	0.373**	0.322***	0.606	0.077	0.000
	[0.055]	[0.065]	[0.050]		[0.067]	[0.148]	[0.111]			
Caterpillar	0.498***	0.235***	0.198***	0.665	0.321***	0.507***	0.352***	0.765	0.001	0.000
	[0.053]	[0.061]	[0.056]		[0.058]	[0.083]	[0.073]			
Chevron	0.556***	0.234***	0.138***	0.687	0.521***	0.295***	0.334***	0.783	0.031	0.000
	[0.054]	[0.069]	[0.052]		[0.095]	[0.111]	[0.063]			
Cisco Systems	0.453***	0.206***	0.229***	0.507	0.529***	0.395**	0.125	0.639	0.063	0.000
	[0.045]	[0.052]	[0.050]		[0.047]	[0.177]	[0.166]			
Coca Cola	0.464***	0.163***	0.219***	0.433	0.507***	0.419**	0.036	0.560	0.001	0.000
	[0.062]	[0.059]	[0.055]		[0.056]	[0.163]	[0.183]			
Disney	0.491***	0.182***	0.217***	0.532	0.413***	0.430**	0.232***	0.657	0.128	0.000
	[0.054]	[0.057]	[0.053]		[0.118]	[0.203]	[0.089]			
Goldman Sachs	0.440***	0.233***	0.231***	0.558	0.433***	0.340***	0.404***	0.662	0.047	0.000
	[0.082]	[0.074]	[0.051]		[0.060]	[0.083]	[0.062]			
Home Depot	0.488***	0.241***	0.176***	0.587	0.535***	0.318***	0.199***	0.698	0.004	0.000
	[0.048]	[0.058]	[0.050]		[0.052]	[0.077]	[0.075]			
Honeywell	0.519***	0.202***	0.196***	0.625	0.483***	0.298***	0.312***	0.741	0.183	0.000
	[0.046]	[0.058]	[0.052]		[0.050]	[0.086]	[0.070]			
IBM	0.504***	0.207***	0.138***	0.493	0.478***	0.513***	0.016	0.630	0.002	0.000
	[0.054]	[0.059]	[0.052]		[0.055]	[0.100]	[0.095]			
Intel	0.458***	0.209***	0.198***	0.480	0.638***	0.263**	0.196*	0.637	0.071	0.000
	[0.059]	[0.056]	[0.044]		[0.071]	[0.113]	[0.111]			
JPMorgan	0.565***	0.164***	0.202***	0.678	0.468***	0.358***	0.324***	0.764	0.233	0.000
	[0.043]	[0.050]	[0.045]		[0.057]	[0.118]	[0.086]			
Johnson Johnson	0.477***	0.213***	0.169***	0.487	0.544***	0.358***	0.122	0.610	0.001	0.000
	[0.056]	[0.061]	[0.058]		[0.065]	[0.130]	[0.146]			
MMM	0.327***	0.238**	0.297***	0.396	0.216	0.771*	0.098	0.547	0.517	0.000
	[0.111]	[0.096]	[0.063]		[0.232]	[0.451]	[0.179]			
McDonalds	0.419***	0.186***	0.258***	0.431	0.427***	0.308**	0.222*	0.529	0.003	0.000
	[0.046]	[0.056]	[0.052]		[0.064]	[0.133]	[0.128]			
Merck	0.418***	0.249***	0.198***	0.474	0.546***	0.403***	0.104	0.580	0.051	0.000
	[0.071]	[0.075]	[0.052]		[0.106]	[0.102]	[0.151]			
Microsoft	0.439***	0.241***	0.173***	0.461	0.450***	0.359***	0.211***	0.614	0.000	0.000
	[0.069]	[0.066]	[0.044]		[0.079]	[0.072]	[0.075]			
Nike	0.519***	0.168***	0.205***	0.548	0.300***	0.562**	0.215	0.704	0.022	0.000
	[0.056]	[0.056]	[0.052]		[0.091]	[0.238]	[0.157]			
Procter Gamble	0.085	0.164*	0.284***	0.050	0.613***	1.465	−0.968	0.205	0.515	0.000
	[0.069]	[0.093]	[0.056]		[0.067]	[1.106]	[0.991]			
Salesforce	0.423***	0.225**	0.235***	0.483	0.649***	0.265**	0.351***	0.611	0.862	0.000
	[0.074]	[0.103]	[0.075]		[0.088]	[0.106]	[0.088]			
Travelers	0.452***	0.261***	0.182***	0.560	0.438***	0.423***	0.149	0.657	0.002	0.000
	[0.055]	[0.060]	[0.053]		[0.049]	[0.119]	[0.098]			
UnitedHealth	0.281***	0.264***	0.345***	0.431	0.528***	0.495***	0.134	0.575	0.042	0.000
	[0.093]	[0.075]	[0.067]		[0.124]	[0.152]	[0.199]			
Verizon	0.379***	0.286***	0.145***	0.381	0.459***	0.372***	0.140*	0.459	0.247	0.000
	[0.044]	[0.058]	[0.055]		[0.060]	[0.066]	[0.077]			
Visa	0.351***	0.263***	0.249***	0.412	0.569***	0.274***	0.305***	0.528	0.045	0.000
	[0.052]	[0.063]	[0.055]		[0.113]	[0.105]	[0.118]			
Walmart	0.340***	0.291***	0.188***	0.359	0.471***	0.409***	0.105	0.468	0.002	0.000
	[0.054]	[0.065]	[0.058]		[0.090]	[0.077]	[0.098]			

Table 2: In-Sample Results: CNN-HAR Models.
The table shows estimation results for the CNN-HAR model in equation (3) at the stock level, where the CNN components have been fitted on the training data set. Robust standard errors are given in parentheses. ***, **, *** indicate significance at the 1%, 5%, and 10% significance level respectively.

Stock	$\tilde{\beta}_i^{(d)}$	$\tilde{\beta}_i^{(w)}$	$\tilde{\beta}_i^{(m)}$	$\tilde{\gamma}_i^{(d)}$	$\tilde{\gamma}_i^{(w)}$	$\tilde{\gamma}_i^{(m)}$	N	R^2
American Express	−0.284***	−0.097**	−0.092**	0.691***	0.478***	0.432***	1740	0.751
	[0.054]	[0.045]	[0.041]	[0.074]	[0.077]	[0.061]		
Amgen	−0.149***	0.047	−0.016	0.625***	0.253***	0.430***	1740	0.661
	[0.051]	[0.047]	[0.04]	[0.079]	[0.085]	[0.08]		
Apple	−0.191	−0.223*	−0.125**	0.779***	0.592**	0.333	1740	0.627
	[0.191]	[0.114]	[0.057]	[0.245]	[0.298]	[0.206]		
Boeing	−0.129*	−0.168**	−0.006	0.496***	0.528***	0.399***	1740	0.613
	[0.075]	[0.078]	[0.045]	[0.087]	[0.191]	[0.115]		
Caterpillar	−0.247***	−0.07	0.003	0.597***	0.544***	0.414***	1740	0.772
	[0.065]	[0.05]	[0.043]	[0.098]	[0.087]	[0.078]		
Chevron	−0.342**	−0.066	−0.119**	0.810***	0.456***	0.468***	1740	0.796
	[0.139]	[0.063]	[0.047]	[0.173]	[0.128]	[0.073]		
Cisco Systems	−0.296**	−0.146	−0.049	0.763***	0.590**	0.224**	1740	0.657
	[0.124]	[0.101]	[0.042]	[0.116]	[0.246]	[0.112]		
Coca Cola	−0.166	−0.334***	0.053	0.657***	0.628***	0.152	1740	0.579
	[0.123]	[0.082]	[0.056]	[0.145]	[0.137]	[0.104]		
Disney	−0.166	−0.266**	−0.147**	0.494***	0.703**	0.454***	1740	0.677
	[0.105]	[0.134]	[0.07]	[0.132]	[0.294]	[0.089]		
Goldman Sachs	−0.071	−0.084	−0.029	0.474***	0.424***	0.488***	1740	0.663
	[0.047]	[0.052]	[0.042]	[0.087]	[0.101]	[0.072]		
Home Depot	−0.201***	−0.093	−0.017	0.712***	0.398***	0.277***	1740	0.705
	[0.076]	[0.066]	[0.039]	[0.1]	[0.098]	[0.062]		
Honeywell	−0.223**	−0.073	−0.065	0.661***	0.379***	0.435***	1740	0.749
	[0.105]	[0.065]	[0.051]	[0.117]	[0.13]	[0.081]		
IBM	−0.274**	−0.184***	−0.090*	0.727***	0.636***	0.225***	1740	0.649
	[0.109]	[0.065]	[0.049]	[0.116]	[0.11]	[0.074]		
Intel	−0.180**	−0.037	−0.106**	0.780***	0.298**	0.357***	1740	0.646
	[0.087]	[0.069]	[0.049]	[0.122]	[0.137]	[0.089]		
JPMorgan	−0.064	−0.105*	0.028	0.515***	0.438***	0.363***	1740	0.766
	[0.059]	[0.06]	[0.04]	[0.087]	[0.134]	[0.082]		
Johnson Johnson	−0.113	−0.238***	−0.089	0.582***	0.561***	0.328***	1740	0.622
	[0.116]	[0.066]	[0.07]	[0.136]	[0.126]	[0.112]		
MMM	−0.047	−0.324	−0.135	0.125	1.151*	0.302***	1740	0.567
	[0.091]	[0.22]	[0.108]	[0.298]	[0.675]	[0.107]		
McDonalds	−0.194	−0.234***	−0.041	0.609***	0.452***	0.365***	1740	0.541
	[0.147]	[0.07]	[0.059]	[0.162]	[0.113]	[0.078]		
Merck	−0.067	−0.135**	−0.046	0.594***	0.518***	0.197	1740	0.583
	[0.166]	[0.06]	[0.048]	[0.229]	[0.118]	[0.158]		
Microsoft	−0.159**	−0.125**	−0.064	0.524***	0.492***	0.348***	1740	0.623
	[0.065]	[0.05]	[0.044]	[0.079]	[0.079]	[0.079]		
Nike	−0.114	−0.243**	−0.026	0.380***	0.694***	0.386***	1740	0.718
	[0.075]	[0.097]	[0.041]	[0.122]	[0.268]	[0.124]		
Procter Gamble	0.008	−0.21	−0.009	0.583***	1.697	−0.928	1740	0.210
	[0.045]	[0.152]	[0.054]	[0.084]	[1.249]	[0.93]		
Salesforce	−0.07	−0.02	0.003	0.731***	0.294**	0.356***	1740	0.611
	[0.11]	[0.111]	[0.063]	[0.156]	[0.134]	[0.081]		
Travelers	−0.297***	−0.196**	−0.014	0.761***	0.556***	0.231***	1740	0.671
	[0.092]	[0.083]	[0.043]	[0.116]	[0.12]	[0.054]		
UnitedHealth	−0.061	−0.183**	0.042	0.529***	0.685***	0.18	1740	0.580
	[0.075]	[0.072]	[0.052]	[0.17]	[0.193]	[0.146]		
Verizon	−0.113	−0.110**	−0.002	0.574***	0.464***	0.183**	1740	0.462
	[0.175]	[0.056]	[0.061]	[0.187]	[0.087]	[0.084]		
Visa	−0.073	−0.148**	−0.061	0.597***	0.436***	0.425***	1740	0.532
	[0.079]	[0.059]	[0.054]	[0.146]	[0.125]	[0.114]		
Walmart	−0.186	−0.242***	0.04	0.644***	0.622***	0.163**	1740	0.481
	[0.14]	[0.069]	[0.051]	[0.168]	[0.101]	[0.075]		

4.2 Out-of-Sample Results

After fitting the models on the training data, we use the test data set to assess the out-of sample performance of the models. Table 3 presents the RMSEs evaluated on the test data set for the CNN-RV, the benchmark HAR (HAR), the combined model (CNN-HAR) as well as for the individual CNN predictors based on daily ($CNN^{(d)}$), weekly ($CNN^{(w)}$), and monthly ($CNN^{(m)}$) intraday return images. Remarkably, for 23 out of our 28 sample stocks, we observe the lowest RMSE for the CNN-RV model. The CNN-RV model thereby not only beats the benchmark HAR model with respect to its forecasting performance, but for most stocks also the combined CNN-HAR model. The latter beats the HAR model for 20 out of 28 stocks. The HAR model shows the lowest RMSE for only 3 out of 28 sample stocks.[5]

Comparing the CNN-RV results with the individual CNN predictors based on daily, weekly and monthly images (Columns 1-3 in table 3) only, reveals that their combination within a HAR-like linear framework further decreases the RMSE. This might be interpreted as evidence for the validity of the heterogeneous market hypothesis, since even the monthly images, which depict the return series at a 22-minute frequency, cannot encompass all the information included within the daily and weekly images, leading to a further prediction improvement when combining them within a linear model, i.e. within the CNN-RV. These results might also indicate different dynamics with respect to daily, weekly, and monthly input images and the next day's RV predictions, which again supports the idea of training three separate CNNs for each time frame and subsequently combine the resulting predictors.

[5]Forecasting performances relying on the mean absolute error (MAE) are given in table A.4 in the appendix and show similar results. For the majority of stocks (21 out of 28) the CNN-RV model exhibits the lowest MAE, followed by the HAR model, which gives the highest predicting accuracy for 7 out of 28 stocks.

Table 3: Out-of-Sample Forecasting Performance based on the RMSE.
The table reports the RMSE of each model's day-ahead forecasts based on the test set. The first three columns present the results of the individual CNNs by themselves. The fourth column presents the RMSE of the CNN-RV model described in equation 2. The fifth denotes the RMSE of the benchmark HAR in equation 1 and the last column the RMSE of the combined CNN-HAR model as given in equation 3.

Stock	RMSE					
	$\text{CNN}^{(d)}$	$\text{CNN}^{(w)}$	$\text{CNN}^{(m)}$	CNN-RV	HAR	CNN-HAR
American Express	23.042	22.584	24.439	21.977	22.835	22.110
Amgen	37.711	35.551	37.864	35.712	36.308	36.171
Apple	33.548	32.677	34.520	32.406	33.167	32.408
Boeing	40.195	39.162	40.015	37.285	38.440	37.388
Caterpillar	35.526	32.712	34.225	31.238	32.199	31.480
Chevron	22.586	22.157	24.339	21.624	21.421	22.276
Cisco Systems	28.799	28.174	30.656	27.429	28.697	27.863
Coca Cola	17.781	16.765	18.166	16.424	16.835	16.828
Disney	26.975	27.148	28.378	26.117	25.950	26.701
Goldman Sachs	27.271	25.808	28.168	24.872	24.922	25.178
Home Depot	26.551	25.635	28.354	25.035	25.597	25.172
Honeywell	23.519	23.574	23.752	22.497	22.985	22.626
IBM	28.078	28.344	30.675	27.229	28.900	27.280
Intel	31.476	29.885	30.941	29.587	29.598	29.832
JPMorgan	22.579	22.711	24.275	21.890	22.208	22.011
Johnson Johnson	32.657	32.523	33.615	31.924	32.372	32.246
MMM	29.940	28.846	30.086	28.570	29.191	28.904
McDonalds	21.260	20.830	22.036	20.186	20.501	20.447
Merck	25.411	25.275	25.816	24.367	24.218	24.561
Microsoft	29.763	29.436	31.000	28.086	28.446	28.849
Nike	26.477	25.737	27.813	25.047	25.591	25.425
Procter Gamble	19.334	18.902	20.411	23.595	21.601	24.519
Salesforce	41.178	37.582	40.204	34.045	35.833	34.262
Travelers	20.448	19.908	21.360	19.121	19.424	19.583
UnitedHealth	32.552	33.160	35.471	31.100	32.874	31.728
Verizon	20.020	20.535	22.005	19.186	19.871	19.285
Visa	24.857	24.494	26.249	23.590	24.907	23.927
Walmart	21.825	21.493	22.899	20.731	21.525	20.822
Average	27.549	26.843	28.490	26.103	26.658	26.424

To assess the statistical significance of the difference in prediction accuracy, we conducted Diebold-Mariano tests (Diebold and Mariano, 2002) to test the null hypothesis of no difference in the prediction accuracy between the CNN-RV and the HAR model (see table A.5 in the appendix). The results indicate that based on the RMSE the CNN-RV shows a significantly higher prediction accuracy for 10 (4) out of 28 sample stocks at the 10% (5%) significance level. Relying on the MAE yields similar test results (see table A.5 in the appendix). Figure 5 offers a visualization of the out-of-sample prediction performances for the CNN-RV and CNN-HAR models using the HAR model as a benchmark (inspired by a similar visualisation in Christensen et al. (2021)).

Figure 5: Out-of-Sample Forecasting Performance

The figure reports the RMSE of day-ahead out-of-sample forecasts of the CNN-RV and CNN-HAR models relative to the HAR model's RMSE, i.e. values larger then 1 imply a higher RMSE and worse performance compared to the HAR model and vice versa. Boxplots display the interquartile range for each model. The central mark is the median, while the mean observation is marked with a circle. The whiskers give the outermost observations for that model.

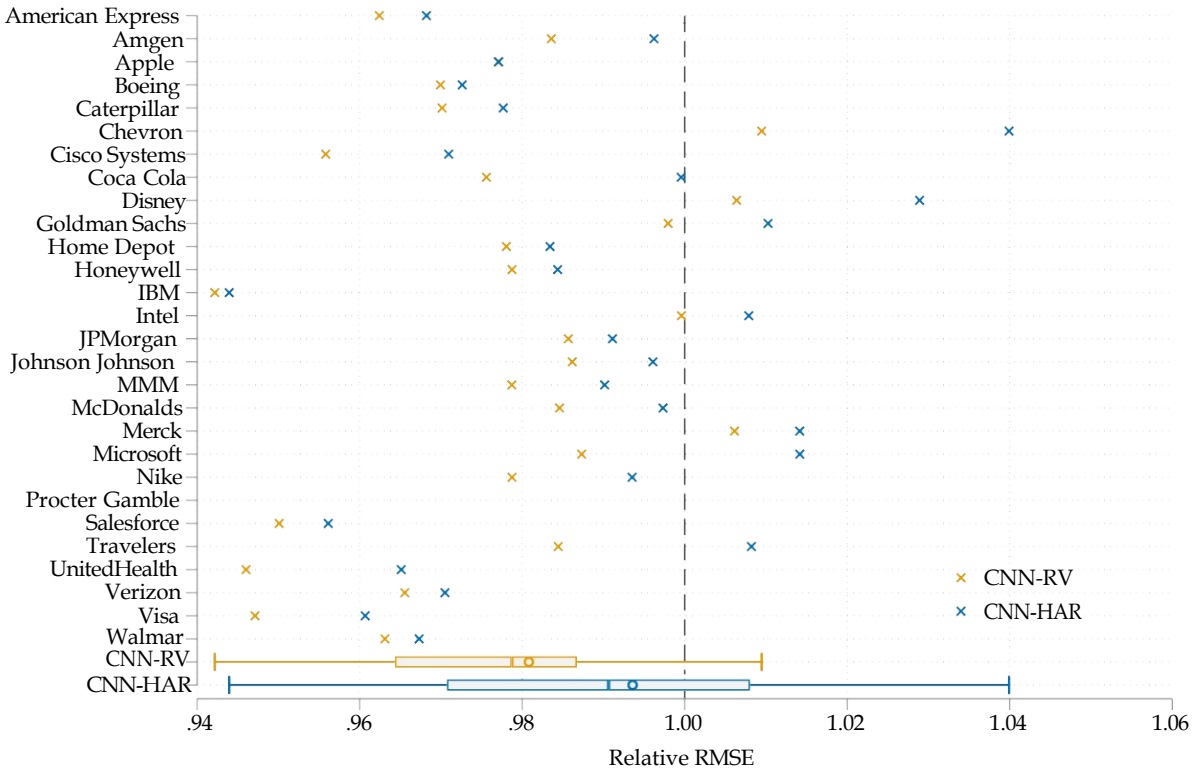


Figure 5 presents the out-of-sample RMSEs for the CNN-HAR and CNN-RV models in comparison to the HAR model's RMSE. Therefore, values below one represent an improvement in out-of-sample forecasting performance above the benchmark HAR model. Furthermore, it displays boxplots showing the interquartile range for each model. The central mark denotes the median, while the mean observation is marked with a circle. The whiskers give the outermost observations for that model. Figure 5 again highlights the gain in prediction accuracy of the CNN-RV model over the benchmark HAR model. It also depicts the overall higher prediction accuracy of the CNN-RV model compared to the combined CNN-HAR approach.

Overall, the results suggest that both, the CNN and the HAR-RV components, contribute significantly to explaining variation in RV. Of course, the inclusion of additional regressors will always improve in-sample fit. However, a detailed analysis of the CNN and HAR-RV components suggests that both contribute significantly to increased model fit. However, for out-of-sample forecasting performance, our results indicate that the CNN-RV model is superior to the CNN-HAR model. This finding suggests that the capacity of deep learning methods to identify intricate patterns when mapping intraday returns to the next day's daily RV results in more accurate forecasts. At the same time, the significance of the individual CNN predictors, i.e. based on daily, weekly and monthly intraday data, highlights the importance of the standard HAR structure when predicting RV.

Combining deep learning predictors based on a cascading structure as underlying the HAR model, provides an example of how deep learning methods can be integrated into traditional frameworks. The model represents one potential trade-off between model flexibility, leading to prediction accuracy, while retaining some interpretability of individual components.

5 Conclusion

We analyze whether merging the worlds of classical volatility forecasting and modern deep learning techniques produces synergies that allow improvement upon existing models. We implement a CNN-RV model, which consists of predictors of daily realized volatility based on convolutional neural nets. We construct three separate CNNs trained on coloured images of intraday returns over the previous day, week, and month to predict day-ahead RV. Although there is no in-depth theoretical justification for the CNN-based model, one could adapt the interpretation of the Heterogeneous Market Hypothesis. Rather than assuming that the partial volatilities at each level of the cascade underlying the HAR model follow an AR(1) process of past RV (Corsi, 2009), volatilities are assumed to follow any arbitrary, linear, or non-linear process. The utilization of CNNs as non-linear function approximators can be understood in this light. Moreover, CNNs can leverage the entire image-transformed intraday return series as a direct input into a volatility forecasting model to extract predictive, non-linear patterns.

Using minutely return data on 28 stocks in the DJIA from 2010 until 2019, we demonstrate that our proposed model surpasses the baseline HAR model in explaining variation in RV and out-of-sample forecasting performance. However, it is worth noting that there are several cases where the CNN and standard HAR components at the same aggregation period are individually significant. This finding demonstrates that the CNNs effectively capture predictive patterns in intraday returns that remain undetected by the HAR model, providing significant *additional* information towards explaining variation in day-ahead RV. Therefore, our findings highlight the potential of incorporating deep learning features

into more traditional forecasting approaches.

Further research could thus pursue various avenues. Firstly, configurations of the model which allow for multi-step-ahead forecasting should be devised. Secondly, a more systematic approach could involve experimenting with various image designs, NN architectures, and hybridization of various classical approaches. Such an approach could yield significant improvements in model performance. Lastly, further research should investigate whether the improved performance of the model above the HAR baseline also translates into economic benefits, for example by evaluating an investments strategy based on the models predictions or risk measures such as Value at Risk.

Appendix

A The HAR Model

Assume that the dynamics of the log price of a stock evolves according to the stochastic differential equation given by

$$dp(t) = \mu(t)dt + \sigma(t)dW(t)$$

$\mu(t)$ denotes the drift, $W(t)$ is a standard Brownian motion, and $\sigma(t)$ gives the spot volatility, which is assumed to be independent of $W(t)$.

Following Corsi (2009) the square root of the daily integrated variance, which is given by

$$IV_t = \int_{t-1}^{t} \sigma^2(\omega)d\omega$$

can be estimated by realized volatility given by the square root of the sum of equally spaced returns over a day

$$RV_t = \sqrt{\sum_{n=0}^{N-1} r_{t,n}^2}$$

where $r_{t,n} = p(t - n\Delta) - p(t - (n + 1)\Delta)$ with $\Delta = 1/N$.

Corsi (2009) then proposes the so-called heterogeneous autoregressive model (HAR) model for forecasting RV_t. The model has a simple autoregressive structure and includes averages of daily realized volatilities over different time horizons. For an aggregation period $j \in \mathbb{N}$, we denote the respective average of daily realized volatilities by

$$RV_t^{(j)} = \frac{1}{j} \sum_{n=0}^{j-1} RV_{t-n}$$

Motivated by the Heterogeneous Market Hypothesis (Müller et al., 1993, Corsi, 2009), the HAR model is an additive cascade model of different partial volatility components. These volatility components are presumed to be generated by the heterogeneous actions of market actors, originating from differences in their time horizons. The model differentiates between short-term traders who operate daily, medium-term traders operating weekly, and long-term actors who adjust their positions monthly. Observing the interrelations between volatility at different time horizons reveals that volatility over longer intervals substantially influences volatility over shorter intervals and vice versa. From this, Corsi (2009) models the actions of short-term traders as dependent on their expectations of longer-term volatility. On the other hand, short-term variations in returns do not affect the trading strategies of longer-term actors. Therefore, Corsi (2009) models the three levels of the volatility cascade as:

$$\begin{aligned}
\tilde{\sigma}_{t+1m}^{(m)} &= c^{(m)} + \phi^{(m)} RV_t^{(m)} + \tilde{\omega}_{t+1m'}^{(m)} \\
\tilde{\sigma}_{t+1w}^{(w)} &= c^{(w)} + \phi^{(w)} RV_t^{(w)} + \gamma^{(w)} \mathbb{E}_t \left[\tilde{\sigma}_{t+1m}^{(m)} \right] + \tilde{\omega}_{t+1w'}^{(w)} \\
\tilde{\sigma}_{t+1d}^{(d)} &= c^{(d)} + \phi^{(d)} RV_t^{(d)} + \gamma^{(d)} \mathbb{E}_t \left[\tilde{\sigma}_{t+1w}^{(w)} \right] + \tilde{\omega}_{t+1d'}^{(d)}
\end{aligned} \tag{A.1}$$

where $\tilde{\sigma}_{t+1d}^{(d)}$, $\tilde{\sigma}_{t+1w}^{(w)}$, and $\tilde{\sigma}_{t+1m}^{(m)}$ are the latent partial volatilities generated by the daily, weekly, and monthly market components. Furthermore, $RV_t^{(d)}$, $RV_t^{(w)}$, and $RV_t^{(m)}$ are, respectively, the daily, weekly, and monthly (ex-post) observed realized volatilities. These multiperiod realized volatilities are defined as simple averages of the daily quantities. The weekly component averages the past 5 and the monthly component averages the

past 22 trading days. Lastly, the volatility innovations $\tilde{\omega}^{(d)}_{t+1d'}$, $\tilde{\omega}^{(w)}_{t+1w'}$ and $\tilde{\omega}^{(m)}_{t+1m'}$ are contemporaneously and serially independent zero-mean nuisance variates. This notation, again, reveals the underlying assumption of Corsi's HAR model that each volatility component in the cascade is determined by a type of market participant who forms his expectations of next-period volatility based on his observation of current RV and their expectation for the longer-term volatility. For the longest, that is, the monthly time scale, only the AR(1) structure remains. Relying on realized volatility (RV) based on squared intraday returns for daily volatility estimation, the weekly and monthly components are derived on weekly and monthly daily RV averages, respectively, resulting in the HAR-RV specification: [6]

$$RV^{(d)}_{t+1d} = c + \beta^{(d)} RV^{(d)}_{t} + \beta^{(w)} RV^{(w)}_{t} + \beta^{(m)} RV^{(m)}_{t} + \omega_{t+1d} \tag{A.2}$$

This final HAR model has a simple autoregressive structure in the RV which can be estimated via Ordinary Least Squares (OLS).

[6] As described in Corsi (2009), recalling that the daily partial volatility component is equal to the integrated volatility yields $IV^{(d)}_{t+1d} = c + \beta^{(d)} RV^{(d)}_{t} + \beta^{(w)} RV^{(w)}_{t} + \beta^{(m)} RV^{(m)}_{t} + \tilde{\omega}_{t+1d}$. Further, notice that ex post $IV^{(d)}_{t+1d}$ can be written as $IV^{(d)}_{t+1d} = RV^{(d)}_{t+1d} + \omega^{(d)}_{t+1d'}$. Substituting this into the equation above yields the final representation of the cascade model.

Additional Tables

Table A.1: Final Layer Tuning Results

Hyperparameters		CNN model		
Dropout	Units	$\text{CNN}^{(d)}$	$\text{CNN}^{(w)}$	$\text{CNN}^{(m)}$
0.10	250	1316.00	1299.49	1362.16
0.10	500	1335.00	1335.08	1392.51
0.10	1000	1299.88	1323.17	1396.11
0.10	5000	1307.63	1314.62	1361.54
0.10	10000	1282.01	1350.78	1399.80
0.25	250	1293.95	1349.51	1374.38
0.25	500	1295.68	1346.92	1390.03
0.25	1000	1288.77	1324.46	1394.87
0.25	5000	1295.17	1335.65	1402.66
0.25	10000	1307.13	1315.79	1401.99
0.50	250	1286.22	1320.14	1365.91
0.50	500	1296.64	1311.84	1396.45
0.50	1000	1283.03	1317.95	1373.00
0.50	5000	1287.96	1315.72	1400.89
0.50	10000	1294.10	1314.36	1401.27
0.75	250	1273.45	1314.76	1386.41
0.75	500	1271.89	1301.87	1360.09
0.75	1000	1268.75	1305.13	1383.32
0.75	5000	1285.88	1315.91	1382.05
0.75	10000	1279.79	1290.22	1399.06
0.90	250	1293.47	1300.03	1425.79
0.90	500	1282.63	1296.27	1386.46
0.90	1000	1288.70	1288.37	1375.20
0.90	5000	1274.22	1269.58	1440.66
0.90	10000	1286.48	1300.21	1393.15

Note: The table reports the minimum validation MSE per hyperparameter combination for each CNN model when evaluated on the validation set. To reduce tuning time, we employ an early stopping algorithm with a patience of four.

Table A.2: Average Validation RMSE per Epoch

Epoch	$\text{CNN}^{(d)}$	$\text{CNN}^{(w)}$	$\text{CNN}^{(m)}$
1	42.235	40.832	44.312
2	41.544	40.672	42.297
3	37.079	37.547	40.389
4	36.928	37.090	38.472
5	35.947	36.489	38.620
6	36.150	36.575	37.986
7	35.787	36.642	37.661
8	36.102	36.689	37.764
9	35.901	36.819	37.558
10	36.307	36.617	37.975
11	36.026	36.776	37.536
12	36.029	36.601	37.627
13	36.303	36.654	37.711
14	35.998	36.522	37.519
15	36.137	36.570	37.529
16	36.182	36.549	37.728
17	36.197	36.673	37.569
18	36.237	36.514	37.729
19	36.318	36.514	37.942
20	36.399	36.929	37.727

Note: The table reports the average RMSE of each CNN ensemble until training epoch 20 when evaluated on the validation set.

Table A.3: Correlation Matrix

	1	2	3	4	5	6
1. $RV_{i,t}^{(d)}$	1.000					
2. $RV_{i,t}^{(w)}$	0.814	1.000				
3. $RV_{i,t}^{(m)}$	0.677	0.844	1.000			
4. $\text{CNN}^{(d)}\left[I_{i,t}^{(d)}\right]$	0.900	0.838	0.737	1.000		
5. $\text{CNN}^{(w)}\left[I_{i,t}^{(w)}\right]$	0.853	0.913	0.806	0.939	1.000	
6. $\text{CNN}^{(m)}\left[I_{i,t}^{(m)}\right]$	0.817	0.890	0.870	0.892	0.941	1.000

Note: The table displays the pearson correlation coefficients between the listed variables. The coefficients were calculated on the entire sample.

Table A.4: Out-of-Sample Forecasting Performance based on the MAE

Stock	MAE					
	$\text{CNN}^{(d)}$	$\text{CNN}^{(w)}$	$\text{CNN}^{(m)}$	CNN	HAR	CNN-HAR
American Express	16.496	16.051	17.939	15.477	16.192	15.689
Amgen	21.335	20.843	21.731	20.570	20.778	20.771
Apple	22.724	23.006	24.805	22.342	23.521	23.021
Boeing	27.390	26.877	27.045	26.144	26.639	26.228
Caterpillar	23.905	22.267	22.739	21.884	22.255	22.154
Chevron	15.755	16.227	18.187	15.677	15.333	16.209
Cisco Systems	19.585	19.778	21.700	19.374	20.261	20.103
Coca Cola	12.021	11.608	12.438	11.097	11.389	11.532
Disney	18.587	18.967	19.634	18.209	17.932	18.801
Goldman Sachs	18.936	18.511	19.995	18.130	17.994	18.492
Home Depot	18.060	17.507	19.242	17.257	17.478	17.324
Honeywell	16.843	17.268	18.041	16.609	16.566	16.907
IBM	17.314	17.730	19.948	16.561	17.669	16.781
Intel	21.931	21.178	21.767	21.018	21.178	21.427
JPMorgan	16.276	16.704	17.833	16.045	16.328	16.226
Johnson Johnson	17.204	17.620	19.289	16.734	16.695	16.950
MMM	19.451	19.492	20.469	19.385	19.127	19.736
McDonalds	13.076	13.518	14.437	12.547	12.957	12.643
Merck	16.550	16.385	17.335	15.734	15.692	16.008
Microsoft	20.698	20.527	22.525	20.225	20.684	21.003
Nike	17.802	17.834	18.963	17.425	17.479	18.129
Procter Gamble	13.351	13.451	14.561	17.289	14.464	18.016
Salesforce	27.767	25.234	26.557	23.685	26.799	23.797
Travelers	13.990	13.891	14.451	13.137	13.365	13.386
UnitedHealth	20.601	20.021	20.660	19.755	21.253	20.120
Verizon	13.383	14.328	15.545	13.101	13.483	13.285
Visa	17.501	17.796	19.406	17.343	18.650	17.852
Walmart	14.089	14.166	14.944	13.222	13.721	13.360
Average	27.549	26.843	28.490	26.103	26.658	26.424

Note: The table reports the MAE of each model's day-ahead forecasts in the test set. The first three columns present the results of the individual CNNs by themselves. The fourth column presents the MAE of the CNN-RV model described in equation 2. The fifth denotes the MAE of the benchmark HAR in equation 1 and the last column the MAE of the combined CNN-HAR model as given in equation **??**.

Table A.5: Diebold Mariano Tests Results

Stock	RMSE		MAE	
	test statistic	p-value	test statistic	p-value
American Express	−1.863	0.063	−2.209	0.028
Amgen	−0.988	0.324	−0.455	0.650
Apple	−0.942	0.347	−1.913	0.057
Boeing	−1.847	0.066	−1.104	0.270
Caterpillar	−1.833	0.068	−0.878	0.380
Chevron	0.474	0.636	1.104	0.270
Cisco Systems	−1.937	0.053	−1.904	0.058
Coca Cola	−1.005	0.315	−1.067	0.286
Disney	0.336	0.737	0.746	0.456
Goldman Sachs	−0.091	0.928	0.338	0.736
Home Depot	−1.293	0.197	−0.727	0.467
Honeywell	−0.898	0.370	0.143	0.887
IBM	−2.387	0.018	−2.729	0.007
Intel	−0.022	0.983	−0.379	0.705
JPMorgan	−0.688	0.492	−0.823	0.411
Johnson Johnson	−0.611	0.542	0.098	0.922
MMM	−1.006	0.315	0.557	0.578
McDonalds	−1.164	0.245	−1.705	0.089
Merck	0.237	0.813	0.110	0.912
Microsoft	−0.576	0.565	−1.342	0.180
Nike	−1.189	0.235	−0.146	0.884
Procter Gamble	1.296	0.196	3.076	0.002
Salesforce	−2.243	0.026	−4.439	0.000
Travelers	−0.932	0.352	−0.855	0.393
UnitedHealth	−1.991	0.047	−2.347	0.019
Verizon	−1.859	0.064	−1.298	0.195
Visa	−1.686	0.093	−2.684	0.008
Walmart	−2.071	0.039	−1.841	0.066

Note: The table displays Diebold Mariano test statistics and p-value based on the root mean squared errors (RMSE) as well as the mean absolute error (MAE).

References

Andersen, T. G., & Teräsvirta, T. (2009). Realized volatility. In *Handbook of financial time series* (pp. 555–575). Springer.

Audrino, F., & Faessler, M. (2023). *Every picture tells a story- the har model based on image recognition* [Working Paper].

Barndorff-Nielsen, O. E., & Shephard, N. (2002). Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *64*(2), 253–280.

Blair, B. J., Poon, S.-H., & Taylor, S. J. (2001). Forecasting sp 100 volatility: The incremental information content of implied volatilities and high-frequency index returns [Forecasting and empirical methods in finance and macroeconomics]. *Journal of Econometrics*, *105*(1), 5–26. https://doi.org/https://doi.org/10.1016/S0304-4076(01)00068-9

Borovykh, A., Bohte, S., & Oosterlee, C. W. (2018). Conditional time series forecasting with convolutional neural networks.

Bucci, A. (2020). Realized Volatility Forecasting with Neural Networks. *Journal of Financial Econometrics*, *18*(3), 502–531. https://doi.org/10.1093/jjfinec/nbaa008

Christensen, K., Siggaard, M., & Veliyev, B. (2021). A machine learning approach to volatility forecasting. *Available at SSRN*.

Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, *7*(2), 174–196.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, *20*(1), 134–144. https://doi.org/10.1198/073500102753410444

Golbayani, P., Wang, D., & Florescu, I. (2020). Application of deep neural networks to assess corporate credit rating. *arXiv preprint arXiv:2003.02334*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, *33*(5), 2223–2273.

Hansen, P. R., & Lunde, A. (2011). Forecasting volatility using high frequency data.

Hong, S. Y., Nolte, I., Taylor, S. J., & Zhao, X. (2021). Volatility Estimation and Forecasts Based on Price Durations*. *Journal of Financial Econometrics*, *21*(1), 106–144.

Jiang, J., Kelly, B. T., & Xiu, D. (2023). (re-) imag (in) ing price trends. *Jounral of Finance*, (forthcoming).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kristjanpoller, W., & Minutolo, M. C. (2015). Gold price volatility: A forecasting approach using the artificial neural network–garch model. *Expert systems with applications*, *42*(20), 7245–7251.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, F.-F., Wu, J., & Gao, R. (2022). *Cs231n convolutional neural networks for visual recognition*. Retrieved July 4, 2022, from https://cs231n.github.io/convolutional-networks/#overview

Liu, L. Y., Patton, A. J., & Sheppard, K. (2015). Does anything beat 5-minute rv? a comparison of realized measures across multiple asset classes. *Journal of Econometrics*, *187*(1), 293–311.

Miura, R., Pichl, L., & Kaizoji, T. (2019). Artificial neural networks for realized volatility prediction in cryptocurrency time series. In H. Lu, H. Tang, & Z. Wang (Eds.), *Advances in neural networks – isnn 2019* (pp. 165–172). Springer International Publishing.

Müller, U., Dacorogna, M. M., Davé, R. D., Olsen, R., Pictet, O., & von Weizsäcker, J. (1997). Volatilities of different time resolutions — analyzing the dynamics of market components [High Frequency Data in Finance, Part 1]. *Journal of Empirical Finance*, *4*(2), 213–239. https://doi.org/https://doi.org/10.1016/S0927-5398(97)00007-8

Müller, U., Dacorogna, M., Davé, R., Olsen, R., Pictet, O., & Ward, J. (1993). Fractals and intrinsic time-a challenge to econometricians, presented in an opening lecture of the xxxixth international conference of the applied econometrics association (aea).

Patton, A. J. (2011). Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, *160*(1), 246–256.

Poon, S.-H., & Granger, C. W. (2003). Forecasting volatility in financial markets: A review. *Journal of economic literature*, *41*(2), 478–539.

Rahimikia, E., & Poon, S.-H. (2020). Machine learning for realised volatility forecasting. *Available at SSRN*, *3707796*.

Reisenhofer, R., Bayer, X., & Hautsch, N. (2022). Harnet: A convolutional neural network for realized volatility forecasting. *arXiv preprint arXiv:2205.07719*.

Sezer, O. B., & Ozbayoglu, A. M. (2018). Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, *70*, 525–538.

Sezer, O. B., & Ozbayoglu, A. M. (2019). Financial trading model with stock bar chart image time series with deep convolutional neural networks. *arXiv preprint arXiv:1903.04610*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

Wang, D., Wang, T., & Florescu, I. (2020). Is image encoding beneficial for deep learning in finance? *IEEE Internet of Things Journal*.

Zhang, C., Pu, X., Cucuringu, M., & Dong, X. (2023). Graph neural networks for forecasting multivariate realized volatility with spillover effects.

Zhu, H., Bai, L., He, L., & Liu, Z. (2023). Forecasting realized volatility with machine learning: Panel data perspective. *Journal of Empirical Finance*, *73*, 251–271. https://doi.org/https://doi.org/10.1016/j.jempfin.2023.07.003