Variance Tolerance Factors For Interpreting Neural Networks

Sichao Li*, Amanda Barnard*
School of Computing
Australian National University
Acton ACT 2601, Australia
Sichao.Li@anu.edu.au
Amanda.S.Barnard@anu.edu.au

Abstract

Black box models only provide results for deep learning tasks, and lack informative details about how these results were obtained. In this paper, we propose a general theory that defines a variance tolerance factor (VTF) to interpret the neural networks by ranking the importance of features and construct a novel architecture consisting of a base model and feature model to demonstrate its utility. Two feature importance ranking methods and a feature selection method based on the VTF are created. A thorough evaluation on synthetic, benchmark and real datasets is provided.

Introduction

Machine learning is a powerful tool that can learn and describe the patterns behind data, enabling researchers to make predictions and improve decisions (Hastie, Tibshirani, and Friedman 2009). This advanced technology has been widely employed in scientific disciplines, e.g. psychology, physics and genomics, but many problems are solved using box machine learning models and big datasets that are not representative (Rudin 2019). Knowing the prediction results or a single metric, such as classification accuracy, is insufficient for describing tasks (Doshi-Velez and Kim 2017) and inappropriate for studies in the sciences where predictions must be accountable. For example, the effectiveness of a medical treatment for a patient may be predictable, but how and why the prediction was made is also an important part of the patient record. Interpretability and unambiguous explanations of the model can be crucial.

The definition of interpretability is hard to be established. Miller (Miller 2019) and Kim (Kim, Khanna, and Koyejo 2016) described the interpretability “the degree to which a human can comprehend the cause of a decision”. Humans can understand the behavior and predictions of machine learning by using transparent methods or interpretable methods. Feature importance rankings that measure the contribution of the input variables to the prediction have become an important tool in interpretable machine learning (Samek, Wiegand, and Müller 2017). These ranking methods are often associated with feature selection, as features can be selected or eliminated based on the rankings to provide an optimal subset (Guyon and Elisseeff 2003). Intuitively, if a model can “tell” which features are important to itself, the machine learning system would benefit from a subset of optimal features by reducing the dimensionality (improving efficiency), improving the generalizability of the learning system (Guyon et al. 2002; Song et al. 2007; Wojtas and Chen 2020) and simplifying accountability. These are the primary motivations of this study.

In this paper, we defined the variance tolerance factor (VTF) that evaluates model’s tolerance to features’ variance, and proposed a novel model-agnostic framework based on it to address the lack of interpretability in neural networks (Ibrahim 2013; Olden, Joy, and Death 2004), where the contribution of the input variables (features) in predicting the output (target) is opaque. Two model-agnostic methods are devised: one from data-driven perspective and one based on mathematical principles. Our architecture consists of a “base model” and a “feature model”, where the base model can be constructed in any format based on the type of task (classification or regression). The two models are trained sequentially and the feature model is systematically re-trained until it achieves the same performance as the base task. After training, the base model is fixed and the weights of a mask layer in the feature model provides the general feature importance recognised by the model. These weights can be further processed as the basis for fast feature selection and interpretable ranking methods using domain knowledge. Each method is thoroughly evaluated on synthetic, benchmark and real datasets via representative machine learning models (e.g. Linear Regression, Logistic Regression, Multi-layer Perceptron, Convolutional Neural Network and Recurrent Neural Network) and the study demonstrates the flexibility and interpretability of our framework.

Related Work

Interpretable machine learning methods are normally categorized as model-specific interpretation methods and model-agnostic interpretation methods (Molnar 2018). It’s easy to achieve interpretability by using algorithms that create interpretable models, e.g. linear regression, logistic regression and decision trees. The great advantage of model-agnostic interpretation methods that separate the explanations from the machine learning mode is their flexibility (Liu, Wang, and Matwin 2018). Model-agnostic interpretation methods can be further categorized as global interpretation meth-
ods and local interpretation methods. Global methods describe the average behavior of a machine learning model, such as permutation feature importance (Ho 1995), the partial dependence plot (Friedman 2001), functional decomposition and global surrogate. In contrast, local interpretation methods explain individual predictions. Shapley values from game theory are an example of a method used to explain individual predictions (Lundberg and Lee 2017). These methods report models’ interpretations through feature importance ranking. The importance of features are defined differently based on various algorithm: filter-based methods, e.g. SPEC (Gallagher et al. 2007) and Fisher Score (Gu, Li, and Han 2012), feature discriminative capability, e.g. ReliefF (Robnik-ˇSikonja and Kononenko 2003) and statistics-based methods, e.g. T-Score (Davis and Sampson 1986). The wrapper-based methods depend of the accuracy of specific models and evaluate all the possible combinations of features against the evaluation criterion, which usually suffer the problem of high computation complexity (Tang, Alelyani, and Liu 2014). Embedded methods are model-specific and use their own built-in feature selection methods, similar to model-specific methods discussed above. Learning attention used to indicate the feature importance is proved successful in the past (Gui, Ge, and Hu 2019).

**Theorems and methods**

**Variance Tolerance Factor (VTF)** VFT measures the sensitivity of a model to changes in the features space, by scaling features simultaneously and quantifying the impact without influencing the model performance. If the model performance is unaffected by large scaling factors, the model can’t tolerate much variance in such features and then these features are consequentially important. Thus, the VTF is quantified as the difference between the scaling factor by which multiply and not affect performance and unchanged feature weight, 1, expressed as \( t_{w_{X_i}} = \| w_{X_i} - 1 \| \), where \( t \) is the notation of VTF and \( w_{X_i} \) is the scaling factor. If the VFT tends to 0, the feature importance tends to 1; i.e. the model is less tolerant to change in this dimension.

**Feature Selection** Based on the VTF, unimportant features can be selected such that the deviation between the factor of a feature and 1 is greater than 1, \( t_{w_{X_i}} > 1 \), and eliminated in linear time, \( O(n) \). Specific tasks might apply different thresholds rather than 1.

\[
F_{all} = \{ \text{All Features} \} \\
F_{unimportant} = \forall x F_{all}(t_{w_{X_i}} > 1)
\]

**Auxiliary mask** To find the VTF, an auxiliary mask \( m \in \mathbb{R}^d \) is introduced, which enables the feature set \( \chi \in \mathbb{R}^{n \times d} \) to achieve the same performance on the same model \( Q \), where \( m = \{ w_{X_1}, w_{X_2}, \ldots w_{X_d} \} \) and \( \chi = \{ X_1, X_2, \ldots X_d \} \) is a dictionary of features. Then the following performance equivalence equation can be obtained:

\[
\mathbb{P}(Q(\chi)) \simeq \mathbb{P}(Q(m \odot \chi))
\]

where \( \odot \) denotes the element-wise product and \( \mathbb{P}(\cdot) \) is a function of performance of a machine learning model.

**Problem Formulation**

The problem then becomes ranking the importance of features in \( \chi \) through an auxiliary mask \( m \). Suppose \( \chi \) is a feature set of \( d \) features and \( y \) is the corresponding target set. The performance equivalence equation Eqn.2 can be solved as an optimization problem. The performance function can be substituted by a loss function, as performance of a model is usually quantified by loss. Thus, we can transform the equation as follows:

\[
L_{E'} = \text{argmin}_m (f_{Q_0}(m \odot \chi), y) + \lambda R(\theta) \\
\text{s.t. } L_{E'} \leq \varepsilon
\]

where \( \varepsilon \) is the loss value after optimizing \( L_E = \text{argmin}_\theta (f_Q(\chi), y), f_Q(\cdot) \) is a function of model \( Q \) with parameters \( \theta \). \( R(\cdot) \) is an L2-norm that helps to speed up the optimization process and \( \lambda \) controls the strength of penalty.

**Model Description**

To obtain the mask \( m \) stated in Eqn.3, we propose a framework consisting a base model and a feature model, as shown in Fig.1. The base model is selected and employed to accomplish a supervised learning task, e.g., classification or regression, while the feature model is designated to discover the feature importance through a systematic retraining process. The feature model follows the trained base model and fixes the model parameters, while adding an extra mask layer to learn. The feature model is retrained to achieve the loss retrieved from base model and the weights of mask layer are extracted as the output.

Due to the randomness of weights during initialisation, learning rate and other factors, the value of \( m \) is contingent to a certain extent. Weights can also be misleading with regards to performance. In the case where an unimportant feature is assigned to a high weight, the overall performance is not affected because its contribution to the outcome is virtually insignificant. The weight list in \( m \) is the final parameter used for the feature selection and feature importance ranking.

![Figure 1: The structure of the framework](image-url)
• Each feature equally contributes to the model.
• The weight is randomly chosen so that each iteration commences differently.

In summation, the initial weights are distributed as $U(-0.05, 0.05)$, so that the contribution of each feature is not identical but minor differences can be ignored and the final weight accumulates the effect of all connection weights across all hidden neurons in the network.

Two model-agnostic methods are devised based on VFT for interpreting the model with domain knowledge.

### Recursive Variance Tolerance Weight (RVTW)

From a VFT point of view, the scaling factors (weights) of the important features $w_{X_i}$ tends to 1 as the model restricts the change of the feature instances in training at global level. On the other hand, important features statistically generate greater weights, for example in linear regression, where increasing the value of a feature by one unit changes the estimated outcome by its weight.

To be more convincing, we use a data processing strategy that averages a list of weights from repeatedly retraining of the feature model until the average of the weights reaches a stable point. The weight is also scaled with its standard error to mitigate the uncertainty, because if the model performance is unaffected by large variations in the scaling factors, then the feature is consequentially unimportant.

$$v_{X_i} = \frac{\text{mean}(w_{X_i})}{\text{mean}(t_{w_{X_i}}) e_{w_{X_i}}} = \frac{\frac{1}{N} \sum_{j=0}^{N} w_{X_{ij}}}{\frac{1}{N} \sum_{j=0}^{N} t_{w_{X_{ij}}}}$$

(4)

where the value of $v_{X_i}$ is the measurement of feature importance, $N$ is the number of training times, $w_{X_{ij}}$ is the coefficient of the feature $X_i$ at training time $j$. $e_{w_{X_i}}$ is the standard error $STD(w_{X_i})$.

### Contribution Factor (CF)

Considering the equivalence of Eqn.2, another way to approach $m$ is to solve this as an equation system. For any feature set $\chi$ ($d \geq 1$) contributing to a model, the contribution can be accumulated, such that:

$$C(Q, \chi) = \sum_{i=0}^{d} C(Q, X_i)$$

(5)

where $C(\cdot)$ is a contribution function.

The idea of CF is to treat each feature contribution as an unknown in an equation and construct a solvable equation system. After the feature retraining, we can immediately derive Eqn.6 based on Eqn.5. A factor $\mu$ is introduced to represent the relationship between the weight $w$ and the contribution $C(X)$, and can be derived through experiments. The general case is formulated in Eqn.7. A detailed proof of algorithm is shown in appendix Derive of Contribution Factor.

$$C(X_1) + C(X_2) + \ldots + C(X_i) = C(w_{X_1}, X_1) + C(w_{X_2}, X_2) + \ldots + C(w_{X_i}, X_i)$$

(6)

To construct the equation system the number of retrain attempts must be equal or greater than the number of features, which might result in an overdetermined system. Theoretically, it is enough to solve unknowns in an equation system, given the exact same number of equations, but we need more equations to reduce the effect of bias introduced from unimportant features (discussed above), so that with $d$ unknowns and $N$ equations: ($N > d$). Matrix reduction is necessary to solve this overdetermined system and Gaussian elimination can be applied. With reduced row echelon form, the equation system can be solved, and the contribution of each feature to the performance can be derived.

$$\begin{align*}
\mu_1 C(X_1) + \mu_2 C(X_2) + \ldots + \mu_i C(X_i) &= 1 \\
\vdots \\
\mu'_1 C(X_1) + \mu'_2 C(X_2) + \ldots + \mu'_i C(X_i) &= 1
\end{align*}$$

(7)

### Proof of Concept

To demonstrate the basic feasibility and reliability of the theory, a linear regressor and logistic classifier are selected as a baseline test. The framework is applied to benchmarks and a real-world dataset to evaluate it against practical problems. All datasets used in herein are publicly available and were split into 80% training set and 20% testing set prior to optimization with the same random seed. All models and experiments are conducted using TensorFlow software and their details are tabulated to provide the greatest opportunity for experimental control and reproducibility. Training results are tabulated, or included in learning curves which also quantify any under-fitting or over-fitting. Feature importances are ranked using histograms to give intuitive insights of their effects. To test the reliability and accuracy of the proposed methods, we compare their performance with other well-established methods:

#### Model-agnostic methods:
- Fisher Score: selects features based on similarities.
- ReliefF: selects features based on the identification of near-hit and near-miss instances.
- Permutation method: gives another feature importance profile by measuring the increase in the base model prediction error after permuting a feature.
- AFS: attention-based supervised feature selection.
- KernelShap: computes the Shapley values that fairly distributes the prediction to each individual feature.

#### Model-specific methods:
- Random Forest (RF): regressor provides feature importance based on Gini impurity.
- Connection weights: calculates the sum of products over all hidden layers and ranks the relative importance by the outputs.

### Evaluation Protocols

To evaluate the performance of feature selection method, the traditional recursive feature elimination (RFE) method is applied to the same model and dataset, with the same feature importance rankings.
Unimportant features are selected according to their importance by percentage, from 10% to 90% and a new model independent from previous training and outputs is used to fit the selected subset. The loss or accuracy of the feature subset indicates the importance of dropped features as the performance metric. For MNIST dataset, we followed the evaluation protocols of (Gui, Ge, and Hu 2019) to provide a fair comparison.

Linear Regression and Logistic Regression

The dataset used in the linear regression is The Boston Housing Dataset (Asuncion and Newman 2007), which was collected in 1978 and contains information about houses from various suburbs in Boston, Massachusetts. The dataset can be accessed from the scikit-learn library (Pedregosa et al. 2011), including 506 features and 13 feature variables. The dataset used in the logistic regression is Breast Cancer Wisconsin (Diagnostic) Dataset (Wolberg, Street, and Mangasarian 1992), which was collected in 1955 and contains information about characteristics of the cell nuclei present in the image of a fine needle aspirate (FNA) of a breast mass. The dataset can be accessed from the scikit-learn library (Pedregosa et al. 2011), including 569 samples and 30 features.

A linear regressor and a logistic classifier are built to model the relationship between independent variables and one dependent variable. The model and training details are in appendix Table.2. The base model was first trained with all features to make the decision for 1000 epochs, achieving low testing errors and high accuracies, with no over-fitting and no under-fitting, shown in appendix Fig.2.

Baseline Recursive Variance Tolerance Weight

Based on the outputs of the base model, including its structure and weights, the feature model was constructed with the exact same settings except for one mask layer on the top of the base model. Then the feature model was trained and optimised using all features to predict the target until the loss achieved the same number as base model, which training and details are summarized in appendix Fig.2. According to the theory, the retraining process was conducted numerous times and visualised in the learning curve to provide an insight into the process, from which it is noted that to achieve the same performance the feature model requires a relatively small number of epochs, illustrated in appendix Fig.2. The linear regressor achieves the loss target in 175 epochs and the logistic regressor achieves the loss target in 300 epochs with no over-fitting and no under-fitting.

In both cases, the feature model is retrained 250 times and the the weight information (weight and variance) retrieved from training iterations are visualized in a weight plot in appendix Fig.1. The feature importance is ranked based on the RVTW and to compare the difference with the effect of the VTF, we applied both methods in the feature set, resulting a feature importance profile and a feature unimportance profile, as depicted in the histogram in Fig.3. The feature importance profile varies as the number of retrain times increasing and becomes stable after some iterations, shown in appendix Fig.4.

Baseline Contribution Factor

Given the results from feature model retraining, an equation system was constructed as described in Eqn.7, where \( \mu \) remains unknown. To obtain this parameter, the base model was used to evaluate various settings of the feature set according to the procedure in appendix and the loss change was then used to simulate the relationship between weights and contributions \( \mu \).

Two equation systems of more than 13 equations with 13 unknowns, and more than 30 equations with 30 unknowns were solved with Gaussian reduction algorithm, where the unknowns represent the contribution of each single feature in the feature set. To capture the relationship of the distribution of each feature, the results of the equation system are divided by one of the features’ distribution. Fig.3 (c) presents an overview of the contribution of each feature and detailed contributions are also provided in appendix Fig.5. With the more equations included, the change in the contribution distribution is shown in appendix Fig.6.

Baseline Evaluation

From Table.1, it shows that our feature selection method can achieve similar performance in much shorter time. The permutation method, connection weights, RF, KernelSHAP, Fisher Score, ReliefF and AFS are used as a benchmark to evaluate the performance of the theory applied in the linear regressor. All feature importance profiles are provided in appendix Fig.7. From the importance profile, we can see that LSTAT and RM occurs most among important features in all rankings, and AGE and NOX occurs most among least features in all rankings for Boston Housing Dataset. From Fig.2 (a) we can observe that the model performance drops more with 80% and 90% important features extracted from RVTW and CF than others except RF, while our methods still accurately detect the important features. In the case of 70% important features removed, CF outperforms others and from 60% important features removal, all methods perform similarly. In terms of logistic regression case, mean perimeter, worst area and worst concave points occurs most among important features in all rankings, and symmetry error, worst error and mean fractal dimension occurs most among least features in all rankings. Fig.2 (b) shows that accuracy gaps between baseline and results from RVTW are wider than most methods, while CF performs in average.

Application

The framework is also applied to widely-used neural networks, convolutional neural network (CNN), recurrent neural network (RNN) and multilayer perceptron (MLP). Training details and network structures are tabulated in appendix Table.10.

Convolutional Neural Network

To demonstrate the interpretability of the methods via visual inspection, we employ a CNN to classify the MNIST well-known benchmark dataset (Deng 2012), focusing on the hard-to-distinguish digits “3” and “8”. The subset is a dataset of 13966 \( 28 \times 28 \) grayscale images, and can be accessed from Tensorflow library (Abadi et al. 2015), with 784 features and 2 classes, where each pixel can be treated as...
Figure 2: Feature selection methods’ reliability and accuracy comparison using dataset (a) Boston house price and (b) Breast Cancer Wisconsin (c) MNIST. In the first two datasets, the percentage of selected unimportant features are set from 10% to 90% with an interval of 10%, while the numbers of selected features are set from 3 to 295 with an interval of 10 in MNIST, following the protocol (Gui, Ge, and Hu 2019).

Figure 3: The feature importance rankings of Boston house price (left column) and Breast Cancer Wisconsin (right column) measured by (a) VTF (importance from right to left), (b) RVTW (importance from left to right) and (c) CF (importance from left to right).

Figure 4: Feature importance maps of MNIST generated from different methods. (a) Permutation method, (b) Random forest (RF), (c) Contribution factor, (d) AFS, (e) RVTW, (f) Fisher Score, (g) ReliefF, (h) Unimportant feature profile.

The feature model was trained and optimised using all images to classify the output until the loss achieved the same number as base model. The retraining process was conducted 800 times, which is greater than the number of features (784) needed for the CF. The feature model also achieves the same loss with 100% accuracy. The feature importance is ranked based on RVTW and CF, the pixel importances as shown in Fig.4. Fig.4(c) presents an overview of the contribution of each feature.

The AFS, Fisher Score, ReliefF, permutation method and RF are used as a benchmark. The permutation method results in a feature importance map after permuting pixels across all images in Fig.4(a). The RF regressor fitted the relationship between 784 features and the class, with 98.89% accuracy, and the feature importance based on Gini impurity is ranked in Fig.4(b). We can see from Fig.4(a-g) that the RVTW and ReliefF produce more interpretable and meaningful maps than other methods for human. From the Fig.2(c), we can observe that RVTW achieves the best accuracy on almost all feature selection ranges. The CF method yields a map similar to the permutation method. Unimportant pixels are marked Fig.4(h).
Recurrent Neural Network

To demonstrate the interpretability of the methods in sentiment analysis, we employ the RVTW and CF in a RNN on IMDB movie reviews (Maas et al. 2011), to classify the overall review as positive or negative, where each word is considered as a feature. For the purpose of demonstration, the dataset is loaded from TensorFlow library (Abadi et al. 2015), where each review is encoded as a list of integers. For convenience, only the top 2,000 most common words are considered and actual words are indexed from 3. An RNN base model is constructed following the structure and trained for 500 epochs, achieving 88.24% accuracy with low loss.

The feature model was constructed based on the outputs of the base model, however, the length of sentence is fixed and a sentence can’t contain all words. To update the weight of each word in each iteration and rank all words in a library, the mask layer is located between a one-hot layer and an LSTM layer. The feature model was retrained 2000 times since 2000 features were needed to demonstrate the CF method. The feature model also achieves the similar loss with 88% accuracy. The feature importance is ranked and visualised in Fig.5(a-e).

As mentioned above, due to the large scale of the word bank, we constructed an equation system with 2000 unknowns and 2000 equations to demonstrate the success in RNN. The contribution of top 30 words are ranked and shown in Fig.5(c), where is impossible to present all words.

Considering the computing complexity, only permutation method and RF are used as a benchmark to evaluate the performance of the VTF-based methods applied to an RNN. To fit all sentences and rank their feature importance, all sentences are encoded using one-hot format and fitted using a RF regressor based on Gini impurity. The accuracy is 82.94% results and importances are illustrated in Fig.5(a). By permuting each word in the library in each sentence, the feature importance profile can be obtained according to their loss change, shown in Fig.5(b).

Semantic analysis using RNN is an example showing the feasibility and flexibility of the framework, but when the corpus (each representing a feature) is large, it becomes impractical. Unimportant feature profile from the VTF treats most of the unimportant features equally, so the CF and RVTW capture the most frequently occurring words, like the permutation method, even if some of them are uninformative.

Multilayer Perceptron and Real-world Case

We finally applied our approach on a real-world dataset, addressing an important but challenging task to predict the formation of highly disordered nanoparticle based on their physics. The dataset used for this study contains 4000 gold (Au) nanoparticles generated using molecular dynamics at various temperatures, growth rates and times, which is available online (Barnard, Amanda; Opletal, George 2019). Atomistic, molecular, topological and crystallographic descriptors are characterised as features and the formation energies are assigned as the target property labels. Based on different types of information about the nanoparticles the features are grouped into 5 different feature sets, referred as Bulk (B), Surface (S), Totals (T), Condensed (C) and Expanded (E). A comprehensive evaluation of different machine learning models on these feature sets has been previously reported (Barnard and Opletal 2020). In the present study T feature set is used to train a MLP and all 5 feature sets are used to train individually linear regressors. The model structures are listed in appendix Table.9. A total of 5 linear regressors reproduced the reported LR results (Barnard and Opletal 2020) in appendix Table.1, while MLP achieved better performance in T set, with a MAE of 0.0282.

According to the number of features, the feature model is retrained from 35 to 150 times. The feature importance is ranked based on the VTF, RVTW and CT, and the feature unimportance and importance maps for the MLP are visualised in Fig.5 (f-h). Feature importance profiles extracted from 5 linear regressors are provided in appendix Fig.8. We can observe our histograms differs from the reported Extra-trees outcomes (Barnard and Opletal 2020), which is to be expected. Different machine learning models trained with same dataset lead to different rankings.

Table 1: Comparison of feature selection methods

|                | Linear regression | RFE Base model |
|----------------|------------------|----------------|
| Time complexity | $O(n)$           | $O(n^2)$       |
| MSE            | 3.4014           | 3.4007         | 3.4003         |
| Logistic regression |                |                |
| Time complexity | $O(n)$           | $O(n^2)$       |
| Loss           | 0.0866           | 0.0834         | 0.0812         |
| Recurrent Neural Network |        |                |
| Time complexity | $O(n)$           | $O(n^2)$       |
| Accuracy       | 99.76%           | -              | 100%           |
|                | 87.96%           | -              | 88%            |

Conclusion

In general, the aim of present work is to provide human-readable interpretability to black box models by measuring the tolerance of the model to the change of features. The present approach is motivated by linear regression (Good-fellow, Bengio, and Courville 2016) and Shapley values, and the way to accomplish the idea comes from transformer (Ji, Heo, and Park 2021), fixing the parameters from the base model and training a feature model. Weights of the mask layer on feature model are completely data-driven feature importances and we proposed two methods based on VTF, RVTW and CF, to endow interpretability at a global level, which can be combined with human understanding. More importantly, other possibilities for interpretability still exist. The universality of this approach is demonstrated by deploying in linear regressor, logistic regressor for general cases; CNN for image-related tasks; RNN for semantic analysis and MLP for real-world applications.

Our method is fast and flexible in model selection, feature selection and feature representation, such as visualising individual words and image pixels in the experiments.
Figure 5: Feature unimportance and importance maps of words (a-e) and Gold dataset (f-h) generated from different methods and models. IMDB dataset: (a) Permutation method, (b) RF, (c) Contribution factor, (d) VFT, (e) RVTW. Gold dataset: (f) Variance Tolerance Factor, (g) RVTW and (h) CF.

It can easily make use of any advanced neural networks to be our component-base model in the framework by separating interpretability from models (Ribeiro, Singh, and Guestrin 2016), meaning it is not limited to certain models that might lose predictive performance. In the language processing case, the random forest classifies the words with accuracy 82.94%, while RNN can achieve 88% accuracy. The feature unimportance profile is not exactly the reverse of important feature profile extracted from VTF, same features ranked at different positions, like \textit{AGE} and \textit{NOX} in Fig. 3. With more factors taken into consideration, the outcome of the RVFW ranking is different from other feature importance profiles, but top important features are always selected, although their positions vary as different approaches being applied. It is not fair to judge the performance of a ranking over another, as the interpretability is subjective, while the performance of feature selection methods based on the result from Fig. 2 demonstrate the reliability and accuracy of our methods. CF provides more detailed interpretability for machine learning models by calculating contribution of each feature through solving equation systems. In addition to ranking the feature importance profile, the method also calculates specific contribution values for features. More importantly, the method opens a new field for interpretable methods for further investigation and the potential for directly measuring the impact of “domain-driven” interventions by researchers.

Several comparative and classic methods are used to provide insights from different perspectives, including model-agnostic methods and model-specific methods: connection weights, random forest, KernelShap, permutation method, AFS, ReliefF and Fisher Score. Olden (Olden, Joy, and Death 2004) found that the connection weight approach achieves the best performance for accurately quantifying variable importance among 8 approaches. The performance of the connection weight approach highly depends on the initial weights used for starting the training process (Ibrahim 2013), whereas the retraining strategy in our method mitigates the dependence on initial weights. Tree-based methods, such as random forests, are interpretable models but fail to deal with linear relationships and are impossible to extend to neural networks, while our new method is extensible to all machine learning methods and data types. Unlike permuting instances for a feature, all instances are set to be 0 to completely eliminate the influence of a feature. The two well-accepted ReliefF and Fisher Score have comparably stable performance, but not accurate as RVFW. AFS performed poorly in MNIST setting and the reason might be the choice of CNN.

Limitations and further work There are other machine learning models that can theoretically be used to extend to this work, such as RF and SVM, and even unsupervised models, implied by the section name. Further application to complex models and huge datasets would help address the scalability, but the above experiments demonstrate the general feasibility and opportunities. As the number of features and model layers increase, it is still feasible to re-train the feature models as suggested, but the computational cost rises correspondingly. Mathematically, the time complexity of a forward pass of a trained MLP depends on the input, number of layers and the size of each layer, and the feature model is trainable as long as the base model can be trained. Discovering the specific relationship between the size, complexity and feature model training will be the topic of follow up papers in the future.
References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; and ... Xiaoliang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Asuncion, A.; and Newman, D. 2007. UCI machine learning repository.

Barnard, A. S.; and Opletal, G. 2020. Selecting machine learning models for metallic nanoparticles. *Nano Futures*, 4(3): 035003.

Barnard, Amanda; Opletal, George. 2019. Gold Nanoparticle Data Set. v1. CSIRO. Data Collection. https://doi.org/10.25919/5d395ef9a4291.

Davis, J. C.; and Sampson, R. J. 1986. *Statistics and data analysis in geology*, volume 646. Wiley New York.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Doshi-Velez, F.; and Kim, B. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Gallagher, K. A.; Liu, Z.-J.; Xiao, M.; Chen, H.; Goldstein, L. J.; Buerk, D. G.; Nedeau, A.; Thom, S. R.; Velazquez, O. C.; et al. 2007. Diabetic impairments in NO-mediated endothelial progenitor cell mobilization and homing are reversed by hypoxia and SDF-1α. *The Journal of clinical investigation*, 117(5): 1249–1259.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. Machine learning basics. *Deep learning*, 1(7): 98–164.

Gu, Q.; Li, Z.; and Han, J. 2012. Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*.

Gui, N.; Ge, D.; and Hu, Z. 2019. AFS: An attention-based mechanism for supervised feature selection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 3705–3713.

Guyon, I.; and Elisseeff, A. 2003. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar): 1157–1182.

Guyon, I.; Weston, J.; Barnhill, S.; and Vapnik, V. 2002. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1): 389–422.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. The elements of statistical learnin. *Cited on*, 33.

Ho, T. K. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, 278–282. IEEE.

Ibrahim, O. 2013. A comparison of methods for assessing the relative importance of input variables in artificial neural networks. *Journal of applied sciences research*, 9(11): 5692–5700.

Ji, M.; Heo, B.; and Park, S. 2021. Show, attend and distill: Knowledge distillation via attention-based feature matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7945–7952.

Kim, B.; Khanna, R.; and Koyejo, O. O. 2016. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29.

Liu, X.; Wang, X.; and Matwin, S. 2018. Improving the interpretability of deep neural networks with knowledge distillation. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 905–912. IEEE.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, 4768–4777.

Maas, A.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 142–150.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267: 1–38.

Molnar, C. 2018. A guide for making black box models explainable. *URL: https://christophm.github.io/interpretable-ml-book*.

Olden, J. D.; Joy, M. K.; and Death, R. G. 2004. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological modelling*, 178(3-4): 389–397.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. “ Why should i trust you?” Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

Robnik-Šikonja, M.; and Kononenko, I. 2003. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine learning*, 53(1): 23–69.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215.

Samek, W.; Wiegand, T.; and Müller, K.-R. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Song, L.; Smola, A.; Gretton, A.; Borgwardt, K. M.; and Bedo, J. 2007. Supervised feature selection via dependence estimation. In *Proceedings of the 24th international conference on Machine learning*, 823–830.
Tang, J.; Alelyani, S.; and Liu, H. 2014. Data classification: algorithms and applications. *Data Mining and Knowledge Discovery Series*, 37–64.

Wojtas, M.; and Chen, K. 2020. Feature importance ranking for deep learning. *arXiv preprint arXiv:2010.08973*.

Wolberg, W. H.; Street, W. N.; and Mangasarian, O. L. 1992. Breast cancer Wisconsin (diagnostic) data set. *UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]*.
Supplementary Material

Appendix

Proof of learning algorithm

If we add an extra mask layer in the base model, the feedforward propagation becomes:

$$
\begin{align*}
    a_j^{l+1} &= a_j^l w_j^{l+1} & \text{if } l = 0 \\
    a_j^{l+1} &= \sigma(\sum_i a_i^l w_{ij}^{l+1} + b_j^{l+1}) & \text{if } l > 0
\end{align*}
$$

(1)

where \( w_{ij}^k \) is weight for node \( j \) in layer \( l \) for incoming node \( i \); \( \sigma \) is the activation function; \( b_j^k \) is bias for node \( j \) in layer \( l \) and \( a_j^0 \) is product sum plus bias (activation) \( j \) in layer \( l \).

An error function \( E(X, \theta) \) defines error between the desired output \( y \) and the calculated output \( \hat{y} \) of the neural network with the parameter \( \theta \) with respect to the weights \( w_{ij}^k \) and biases \( b_j^k \). In the training process, weights are updated according to the gradient descent of each iteration. Based on the learning rate \( \eta \), each iteration of gradient descent updates as,

$$
\theta^{t+1} = \theta^t - \eta \frac{\partial E(X, \theta^t)}{\partial \theta}
$$

(2)

where \( \theta^t \) is parameters of the network at iteration \( t \). To illustrate the calculation details of the model with a mask layer the mean squared error function is selected, defined as

$$
E(X, \theta) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y} - y)^2
$$

(3)

According to the backpropagation algorithm and chain rule, the derivation of gradients is straightforward. By deriving \( \frac{\partial E(X, \theta^t)}{\partial w_{ij}^k} = \frac{1}{N} \sum_{t=1}^{N} \frac{\partial E_d}{\partial w_{ij}^k} \), the general form for all input in \( X \) can be summarized by adding all individual gradients. The individual gradient with respect to a general weight is \( \frac{\partial E_d}{\partial w_{ij}^k} \), where the chain rule applies. The error function partial derivative becomes \( \frac{\partial E_d}{\partial \alpha_j^k} \). The first term is usually called the error, denoted \( \delta_j^k \), and the second term can be simplified as \( \alpha_j^{k-1} \). The first term can be transferred to the following equation by applying the chain rule:

$$
\delta_j^k = \frac{\partial E_d}{\partial \alpha_j^k} = \sum_{l=1}^{r+1} \frac{\partial E_d}{\partial \alpha_j^{k+1}} \frac{\partial \alpha_j^{k+1}}{\partial \alpha_j^k} = \sum_{l=1}^{r+1} \delta_l^{k+1} \frac{\partial \alpha_j^{k+1}}{\partial \alpha_j^k}
$$

(4)

where \( r+1 \) is the number of nodes in the next layer. Because all the weights (except for those between the input layer and the mask layer) are not trainable, they can be considered as constants.

According to the linear relationship shown in Fig.1, the partial derivative between the input layer and the mask layer can be derived as

$$
\frac{\partial E}{\partial \alpha_j^0} = \delta_j^1 \frac{\partial \alpha_j^1}{\partial \alpha_j^0} = \sum_{l=1}^{2} \delta_j^l \frac{\partial \alpha_j^2}{\partial \alpha_j^l} = \sum_{l=1}^{2} \delta_j^l w_{jl}^2 \frac{\partial E}{\partial \alpha_j^l} = \delta_j^2 \sum_{l=1}^{2} \delta_j^l w_{jl}^2
$$

(5)

where \( \delta_j^0 \) is input data directly from \( X \). In this equation Fig.5, we can observe that \( \delta_j^0 \) and \( w_{jl}^2 \) are constant and the only variable is \( \delta_j^2 \). If we trace this down to the final layer \( \delta_j^m \), in which \( m \) is the final layer and the formula defined as \( (\hat{y} - y)g'(\alpha^m) \), where \( g \) is the activation function, the only factor that affects the weights between the input layer and the mask layer is the prediction error. Thus, if a feature is important to the outcome, causing the substantial change in prediction error, the coefficient (weight in the model) of the feature would be updated accordingly. However, the coefficient (weight in the model) of unimportant features will not deviate significantly from the initialization value. Overall, the \( \theta \) accumulates the effect of all connection weights across all hidden neurons.

The effect of the prediction error \( \delta_j \) is same in each iteration to all neurons in the mask layer and the factor that controls the \( \theta \) is the weight \( w_{jl}^2 \) in Fig.5 obtained from the base model. Considering that the weight \( w_{jl}^2 \) is calculated based on all connection weights between its connected neurons in hidden layers, the weights imply the importance of the variance.

Derive of Contribution Factor

Considering the contribution of each feature in the feature set and the linear relationship between mask and feature set, we can derive \( C(Q, \{X_0, X_1, X_2, ..., X_i\}) = C(Q, \{w_{X_0}, X_0, w_{X_1}, X_1, w_{X_2}, X_2, ..., w_{X_i}, X_i\}) \). As the model is fixed, the function can be simplified into the following function.

$$
\sum_{i=0}^{N} C(Q, X_i) = \sum_{i=0}^{N} C(Q, w_{X_i})
$$

(6)
To solve the equation with unknowns, the exact corresponding number of equations have to be provided, offered by retraining the system. The \( w_{X_i} \) in the function also needs to be moved outside the parentheses, where the \( w_{X_i} \) represents the relationship between the weight and the contribution.

**Linear regression case explanation** Thinking of the most straightforward model, a linear regressor, the coefficient of a feature is linear to its contribution, as increasing the weight by one unit changes the estimated outcome by 100%, which doubles the contribution of the feature. So the following function can be derived:

\[
C(X_1) + C(X_2) + \cdots + C(X_i) = w_{X_1}C(X_1) + w_{X_2}C(X_2) + \cdots + w_{X_i}C(X_i)
\]

If the equation system can be solved, then the contribution of each feature to the model performance can be decided.

Fortunately, the previous retrain strategy provides a chance out! Given the number of unknown parameters, we can solve the equation system using the exact corresponding number of equations, offered by retraining the system. The following equation system can be solved easily. By setting the output of each equation to 1, the relationship among all features can be derived. The system can be solved using its associated augmented matrix, as it is empirically more suitable for computer manipulations \((AI) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & \cdots & 1 \\ w_{X_1} & \cdots & w_{X_i} \end{pmatrix} \begin{pmatrix} 1 & 1 \end{pmatrix} \]

\[
\left\{ \begin{array}{l}
C(X_1) + C(X_2) + \cdots + C(X_i) = 1 \\
w_{X_1}C(X_1) + w_{X_2}C(X_2) + \cdots + w_{X_i}C(X_i) = 1 \\
\vdots \\
w_{X_1}C(X_1) + w_{X_2}C(X_2) + \cdots + w_{X_i}C(X_i) = 1
\end{array} \right.
\]

**Introduction and derive of \( \mu \)** However, in practice the model usually is much more complex and the factor \( w_{X_i} \) can not be applied to the contribution directly. To deal with the coefficient in general cases, the model can be run with different input settings to determine the relationship between function \( (\cdot) \) and \( w \) by \( \mu \) experimentally.

For a specific coefficient \( w_{X_0} \) of a feature \( X_0 \), all other coefficients are fixed except for the one that is being calculated and we compare the performance between the model with input \( \chi(X_0, X_1, X_2, \ldots, X_i) \) and \( \chi(w_{X_0}X_0, X_1, X_2, \ldots, X_i) \) by multiplying a mask \((w_{X_0}, \ldots, 1)\) to \( \chi \). The change in \( \Delta P \) between these two outcomes \( \Delta P_{\text{partial}} = \mathbb{P}(\chi(X_0, X_1, \ldots, X_i)) - \mathbb{P}(\chi(w_{X_0}X_0, X_1, \ldots, X_i)) \) from \( X_0 \) is \( w_{X_0} \) and represents the effect of part of the \( X_0 \). The total contribution of feature \( X_0 \) can be found by setting \( \Delta P_{\text{total}} = \mathbb{P}(\chi(X_0, X_1, \ldots, X_i)) - \mathbb{P}(\chi(0 \times X_0, X_1, \ldots, X_i)) \).

According to the \( \Delta P_{\text{partial}} \) and \( \Delta P_{\text{total}} \) above, the \( \mathbb{C}(w_{X_0}X_0) \) can be represented by \( \mu \times \mathbb{C}(X_0) \), where the term \( \mu \) is introduced to represent \( \Delta P_{\text{partial}} \) for simplicity, and the simple linear case can be extended to the general case based on Fig.7. The system can then be solved with row operations from Gaussian elimination, including swapping rows positions, add one row onto another and multiplying a row by a non-zero scaler.

Taking a linear regressor as an example, \( Q = X_1 + X_2 + X_3 \), given an input instance \((0.1, 0.3, 0.6)\), the output is \( 1 = 0.1 + 0.3 + 0.6 \). In which \( X_1(0.1), X_2(0.3) \) and \( X_3(0.6) \) can be seen providing 10%, 30% and 60% contribution to the model \( Q \). However, the contribution is not always obvious in real cases. In this simple case, if we meet the output requirement, the following functions \( \text{Fig.8} \) can be defined based on our method above and we can obtain an augmented matrix coefficient matrix \( (AI) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & \cdots & 1 \\ w_{X_1} & \cdots & w_{X_i} \end{pmatrix} \). The matrix gives a relation between features \( 3 \times C(X_1) = C(X_2); 6 \times C(X_1) = C(X_3) \), which satisfies the contribution distribution.

\[
\left\{ \begin{array}{l}
C(X_1) + C(X_2) + C(X_3) = 1 \\
C(X_1) + 2 \times C(X_2) + 1/2 \times C(X_3) = 1 \\
4 \times C(X_1) + C(X_2) + 1/2 \times C(X_3) = 1
\end{array} \right.
\]

**Linear and Logistic Regression Supporting Information**

Supporting information for linear regression and logistic regression, including linear regressor and logistic regressor training settings Table.2, linear and logistic base model and feature model structures Table.3, the training settings of independent model Table.5, learning curves of the base model (left column) and the feature model (right column) Fig.2, weights distribution of linear and logistic models Fig.3, profile change of linear regression and logistic regression as the number of iterations increases Fig.4, the feature importance profile of Boston house price and Breast Cancer Wisconsin from different feature selection methods 7, important features and detailed contribution of each feature Fig.5 and Fig.6 of linear regression and logistic regression with equations accumulated Fig.6.

---

**Application Supporting Information**

Supporting information for applications, including RNN base model and feature model structures Fig.6, RNN model training settings Table.8, Gold training settings Table.8, results of 5 models trained on different Gold feature sets Table.11, Multilayer perceptron base model and feature model structures Table.1, CNN base model and feature model structures Table.7, CNN model training settings Table.10 and feature unimportance maps from different methods Fig.8.
Figure 2: The learning curve with results of the base model (left column) predicting the target property and the feature model (right column) trained to achieve the same performance, showing the accuracy and generalizability. (a) Linear regression, (b) Logistic regression

Figure 3: The weights of linear regression and logistic regression are displayed in weight plot and the target 1 is marked as the line

Figure 4: Profile change of linear regression and logistic regression as the number of iterations increases

Table 1: Results of 5 linear models trained on Bulk (B), Surface (S), Totals (T), Condensed (C) and Expanded (E), Gold feature sets
Figure 5: Detailed contribution of each feature from CF.

Figure 6: Contribution distribution of linear regression and logistic regression with equations accumulated.

Figure 7: The feature importance profile of Boston house price and Breast Cancer Wisconsin from KernalSHAP, random forest (RF), Connection weights, Permutation method (from left to right) (a) Linear regression, (b) Logistic regression.
| Dataset                     | Model                  | Optimiser | Epoch | Batch size | Loss             |
|----------------------------|------------------------|-----------|-------|------------|------------------|
| **Base model**             | **Boston house price** | **Linear regressor** | ADAM  | 1000       | 10   | MSE             |
| **Feature model**          | **Boston house price** | **Linear regressor** | ADAM  | -          | 10   | MSE             |
| **Base model**             | **Breast Cancer Wisconsin Dataset** | **Logistic classifier** | ADAM  | 1000       | 10   | binary_crossentropy |
| **Feature model**          | **Breast Cancer Wisconsin Dataset** | **Logistic classifier** | ADAM  | -          | 10   | binary_crossentropy |

Table 2: Linear regressor and logistic regressor training settings (seed is 3)

| Layer | Layer type | Layer parameters | Num of trainable parameters | Output size |
|-------|------------|------------------|-----------------------------|-------------|
| Base model | L0 | Input layer | - | 14 | N, 13 |
|        | L1 | Dense layer | - | 14 | 13, 13 |
|        | L2 | Output layer | - | 14 | None, 1 |
| Feature model | L0 | Input layer | - | 0 | N, 13 |
|        | L1 | Mask layer | - | 13 | 13, 13 |
|        | L2 | Dense layer | - | 13 | 13, 13 |
|        | L3 | Output layer | - | 13 | None, 1 |

Table 3: Linear base model and feature model structures

| Layer | Layer type | Layer parameters | Num of trainable parameters | Output size |
|-------|------------|------------------|-----------------------------|-------------|
| Base model | L0 | Input layer | - | 31 | N, 30 |
|        | L1 | Dense layer | sigmoid | 31 | 30, 30 |
|        | L2 | Output layer | - | 31 | 30, 1 |
| Feature model | L0 | Input layer | - | 30 | N, 30 |
|        | L1 | Mask layer | - | 30 | 30, 30 |
|        | L2 | Dense layer | sigmoid | 30 | 30, 30 |
|        | L3 | Output layer | - | 30 | 30, 1 |

Table 4: Logistic base model and feature model structures

| Dataset                     | Model | Optimiser | Epoch | Batch size | Loss             |
|----------------------------|-------|-----------|-------|------------|------------------|
| **Boston house price**     | ANN   | ADAM      | 150   | 100        | MSE              |
| **Breast Cancer Wisconsin Dataset** | ANN | ADAM | 150 | 100 | binary_crossentropy |
| **MNIST**                  | CNN   | ADAM      | 150   | 100        | binary_crossentropy |

Table 5: Independent model training settings (seed is 3)

| Base Model structure | Layer | Layer type | Layer parameters | Num of trainable parameters | Output size |
|----------------------|-------|------------|------------------|-----------------------------|-------------|
| L0 | Embedding layer | - | 64000 | None, 200, 32 |
| L1 | LSTM layer | - | 53200 | None, 100 |
| L2 | Dense layer | sigmoid | 101 | None, 1 |

| Feature Model structure | Layer | Layer type | Layer parameters | Num of trainable parameters | Output size |
|-------------------------|-------|------------|------------------|-----------------------------|-------------|
| L0 | Onehot layer | - | 0 | None, 200, 2000 |
| L1 | Mask layer | - | 2,000 | None, 200, 32 |
| L2 | LSTM layer | - | 53200 | None, 100 |
| L3 | Dense layer | sigmoid | 101 | None, 1 |

Table 6: RNN base model and feature model structures
### Table 7: CNN base model and feature model structures

| Layer | Layer type | Layer parameters | Pool size | Kernel size | Num of trainable parameters | Output size |
|-------|------------|------------------|-----------|-------------|-----------------------------|-------------|
| L0    | Input layer | -                | -         | -           | None, 28, 28, 1             |
| L1    | Convolutional layer | Relu | - | 3x3 | 640 | None, 26, 26, 64 |
| L2    | Convolutional layer | Relu | - | 2x2 | 0 | None, 13, 13, 64 |
| L3    | Max pooling layer | - | 2x2 | - | 0 | None, 11, 11, 32 |
| L4    | Max pooling layer | - | 2x2 | - | 0 | None, 5, 5, 32 |
| L5    | Flatten layer | - | - | - | 0 | None, 800 |
| L6    | Drop out layer | - | 0.5 | - | 0 | None, 800 |
| L7    | Dense layer | Softmax | - | - | 1602 | None, 2 |

**Feature Model structure**

| Layer | Layer type | Layer parameters | Pool size | Kernel size | Num of trainable parameters | Output size |
|-------|------------|------------------|-----------|-------------|-----------------------------|-------------|
| L1    | Mask layer | -                | -         | -           | 784 | None, 28, 28, 1 |
| L2    | Convolutional layer | Relu | - | 3x3 | 0 | None, 26, 26, 64 |
| L3    | Max pooling layer | - | 2x2 | - | 0 | None, 13, 13, 64 |
| L4    | Convolutional layer | Relu | - | 3x3 | 0 | None, 11, 11, 32 |
| L5    | Max pooling layer | - | 2x2 | - | 0 | None, 5, 5, 32 |
| L6    | Flatten layer | - | - | - | 0 | None, 800 |
| L7    | Drop out layer | 0.5 | - | - | 0 | None, 800 |
| L8    | Dense layer | Softmax | - | - | 0 | None, 2 |

### Table 8: RNN model training settings

| Dataset | Model | Optimiser | Epoch | Batch size | Loss |
|---------|-------|-----------|-------|------------|------|
| Base model | IMDB | RNN | ADAM | 500 | 64 | binary_crossentropy |
| Feature model | IMDB | RNN | ADAM | - | 64 | binary_crossentropy |

### Table 9: Multilayer perceptron base model and feature model structures

| Layer | Layer type | Layer parameters | Num of trainable parameters | Output size |
|-------|------------|------------------|-----------------------------|-------------|
| L0    | Input layer | -                | N, 10                       |
| L1    | Dense layer | -                | 1728 10, 32 |
| L2    | Dense layer | -                | 528 10, 16 |
| L3    | Output layer | -                | 17 10, 1 |

**Feature Model structure**

| Layer | Layer type | Layer parameters | Num of trainable parameters | Output size |
|-------|------------|------------------|-----------------------------|-------------|
| L0    | Input layer | -                | 0 N, 53                     |
| L1    | Mask layer | -                | 53 N, 53                    |
| L2    | Output layer | -                | 53 10, 1 |
Figure 8: Feature unimportance maps (left) from VTF and feature importance maps, including RVTW and CF (middle and right) of Gold feature sets on different feature sets. (a) Bulk, (b) Surface, (c) Total, (d) Condensed, (e) Expanded.