Interpreting Deep Neural Networks through Prototype Factorization

Subhajit Das[1], Panpan Xu[2], Zeng Dai[2], Alex Endert[1], Liu Ren[2]
[1] *Georgia Institute of Technology, Altanta, GA, USA*
Email: {das, endert}@gatech.edu
[2]*Bosch Research North America, Sunnyvale, CA, USA*
Email: {panpan.xu, zeng.dai, liu.ren}@us.bosch.com

2020 International Conference on Data Mining Workshops (ICDMW) | 978-1-7281-9012-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICDMW51313.2020.00068

***Abstract*—Typical deep neural networks (DNNs) are complex black-box models and their decision making process can be difficult to comprehend even for experienced machine learning practitioners. Therefore their use could be limited in mission-critical scenarios despite state-of-the-art performance on many challenging ML tasks.**

Through this work, we empower users to interpret DNNs with a *post-hoc* analysis protocol. We propose ProtoFac, an explainable matrix factorization technique that decomposes the latent representations at any selected layer in a pre-trained DNN as a collection of weighted prototypes, which are a small number of exemplars extracted from the original data (*e.g.* image patches, shapelets). Using the factorized weights and prototypes we build a *surrogate model for interpretation* by replacing the corresponding layer in the neural network. We identify a number of desired properties of ProtoFac including *authenticity*, *interpretability*, *simplicity* and propose the optimization objective and training procedure accordingly. The method is *model-agnostic* and can be applied to DNNs with varying architectures. It goes beyond per-sample feature-based explanation by providing prototypes as a condensed set of evidences used by the model for decision making.

We applied ProtoFac to interpret pretrained DNNs for a variety of ML tasks including time series classification on electrocardiograms, and image classification. The result shows that ProtoFac is able to extract meaningful prototypes to explain the models' decisions while truthfully reflects the models' operation. We also evaluated human interpretability through Amazon Mechanical Turk (MTurk), showing that ProtoFac is able to produce interpretable and user-friendly explanations.

***Index Terms*—Matrix Factorization, Explainable AI, Deep Neural Networks.**

I. Introduction

Deep neural networks (DNNs) have shown promising results in various machine learning (ML) tasks including image, time-series and many others [1]–[4]. However, given the complexity of their architecture and the high-dimensional internal state, interpreting these models are extremely challenging. Lack of explanation of such models in many real world use cases, especially in high-stake mission critical situations in medicine, finance, etc. makes them less trustworthy or adaptable for use [5].

To address this challenge, a variety of methods have been developed to obtain post-hoc explanations of pre-trained black-box DNN models. With post-hoc explanation techniques, we can get an improved understanding of a model without incurring changes to it and therefore risking lower prediction accuracy. Examples of such methods include calculating feature attribution [6]–[9] or using interpretable surrogates (*e.g.* linear regression) to locally approximate a model's decision boundary [10]. However, most of the techniques only provide per-instance or local explanations and it is difficult to gain an understanding of the model's behaviour as a whole. To obtain global explanations of DNNs, existing methods interpret the representations captured by each neuron at intermediate layers with activation maximization methods [11] or extract concepts highly correlated with model outputs [12], [13]. ML model developers can use these techniques for validation and debugging purposes.

In this paper we introduce ***ProtoFac***, an *explainable matrix* ***Fac****torization* technique that leverages ***Proto****type learning* to extract *user-friendly* explanations from the activation matrix at intermediate layers of DNNs. Our goal is to obtain a set of prototypes with a set of corresponding weights for each input to explain the behaviour of the model *as a whole*. Prototype learning is a form of case-based reasoning, where the model relies on previous examples similar to the present case to perform prediction or classification [14]. It is a reasoning process used frequently in our everyday life. For example, a lawyer may cite an example from an old trial to explain the proceedings of the current trial and a doctor may rely on records of symptoms from past patients to perform diagnosis for new patients. While a number of DNNs already utilize prototype learning for builtin interpretability [15]–[18], our goal is to leverage the idea for *post-hoc*, *global* explanation of DNNs by using the factorized weights and prototype vectors to build an *interpretation surrogate/surrogate model* to mimic the original model's behaviour: reconstruct the activation matrix at the selected layer and feed it to the downstream network to reproduce the predictions of the original model. We outline a number of desired characteristics of the proposed technique (*i.e.* the desiderata):

Authenticity. A reliable and trustworthy explanation of a DNN should have high fidelity to the underlying model by faithfully representing the operations of the network [19]. To this end, the method should not only mimic the underlying model's output but also accurately reconstruct the latent activation matrix in intermediate layers with weighted combinations of prototype vectors.

Interpretability. To obtain interpretable matrix factorization results, the technique should include non-negative constraints

2375-9259/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDMW51313.2020.00068

to ensure additive, not substractive combination of prototypes. Besides that, each prototype should correspond to a realistic example in the data to be human-understandable.

Simplicity. As the principle of Occam's Razor states, the simplest explanation should be adopted whenever possible. Here it means that the explanation of a model's prediction result should use the least possible number of prototypes.

Model-agnostic. Our goal is to develop a generic method that is applicable to DNNs with varying architectures so that it is flexible for models coming up in the future.

We formulate a novel learning objective for matrix factorization considering the above criteria to obtain a set of prototypes and their corresponding weights for model interpretation. The training procedure uses gradient descent and iteratively projects the prototypes to realistic data samples or segments of data samples (*e.g.* image patches, n-grams and shapelets in time-series).

We conduct experiments on a variety of pretrained DNNs for a wide range of ML tasks including time-series classification on electrocardiograms (ECG) and image classification, demonstrating the general applicability of the proposed method. For each experiment, we report the surrogate model's accuracy with respect to both the oracle prediction generated by the original model and the ground truth labels. To evaluate the transferability of the learned prototypes, we take a hold-out dataset, freeze the prototypes learned previously, train the weights only and report the results. We also report case studies and visualize the prototypes identified by the algorithm. ProtoFac is further compared to non-negative matrix factorization techniques [20], [21] using Frobenious loss as a quality metric. Experiments show that our algorithm produces comparable and sometimes superior factorization results. To evaluate human intepretability of the results, we conduct a crowd-sourced quantitative user study via Amazon Mechanical Turk (MTurk). We ask the subjects to interpret the classification result of a given instance by selecting from a set of candidate prototypes. The result shows that ProtoFac is able to select prototypes that align well with user's intuition or common sense for model interpretation. We also conduct various experiments to study the effects of the hyperparameter settings (*e.g.* the number of prototypes k) and the selection of different layers in a DNN.

Below we summarise the contributions:

- ProtoFac, an explainable matrix factorization technique that leverages prototype learning to obtain post-hoc, model-agnostic interpretations of trained DNNs.
- Experimental results on publicly available time-series, and image data showing that our technique faithfully reflects the behaviour of the original model and successfully retrieves meaningful prototypes to explain the models behaviour.
- Crowd-sourced quantitative user study with results showing the effectiveness of our technique in extracting human interpretable prototypes to explain complex DNNs.

II. Related Work

We seek to help make complex ML models interpretable. In order to do so, there are two main alternatives: (1) use inherently interpretable models, or (2) use post-hoc analysis methods to analyze trained DNN models to render them interpretable [19]. Furthermore, past efforts in post-hoc model interpretation can be categorised as local and global explanation techniques. Local explanation techniques show a model's reasoning process in relation to each data instance. Global explanation techniques aim to provide an understanding of the model's behaviour as whole and analyze what knowledge has been acquired after training.

Intrinsically interpretable models. Models such as decision trees, rule-based models [22], additive models [23], sparse linear models [24] are considered inherently interpretable. Unlike DNNs, these models provide internal components that can be directly inspected and interpreted by the user, *e.g.* probing various branches in a decision tree, or visualizing feature weights in a linear model. Though these approaches provide insightful explanations of ML systems' reasoning process, inherently interpretable approaches usually rely on simpler models which may compromise prediction performance in comparison to state-of-the-art DNNs. Recently, a number of DNN architectures also incorporate interpretable components such as attention modules [25] or prototype layers [15]–[18] for intrinsic interpretability. However, such models may need to perform trade-off between interpretability and model performance in terms of prediction accuracy.

Post-hoc local explanation. Local explanation methods show a pre-trained model's reasoning process in relation to each data instance. One of the most popular post-hoc approaches to explain models is calculating and visualizing feature attributions [6]–[9], [26]–[30]. Feature attributions can be computed by slightly perturbing the input features for each instance to verify how the DNN model's prediction response varies accordingly [7], [31]. It can also be computed by back-propagating through the neural network [6]. Another popular local explanation approach samples the feature space in the neighborhood of an instance to compose an additional training set. The training set is used to build an interpretable local surrogate model that mimics the behaviour of the original model. Using this approach an original model's prediction can be explained by an interpretable model (*e.g.* linear regression) that is easier to inspect [10]. However, local explanation approaches are shown to be inconsistent as the explanation is true for only a specific data instance or its neighbors but not for all the items in the data. Furthermore, it could produce contrasting explanations for two data items from the same class label. It could also suffer from adversarial perturbations [32], [33] and confirmation biases [34]. Besides that, post-hoc local explanation methods require users to manually inspect each data sample to review the model's behaviour instead of showing the model's behaviour as a whole.

Post-hoc global explanation. Global explanation techniques aim at providing an overview of the model's

behaviour instead of focusing on individual instances or local input regions [19]. For DNNs, a particular set of global model explanation techniques focus on understanding the latent representations learned by the neural network through activation maximization techniques [11] which calculate inputs that can maximally activate each individual neurons in intermediate layers in a neural network. On the other hand, concept-based explanations show how the model makes predictions globally by showing relevant concepts [12], [13], [35], [36] that are understandable to humans. For example, the technique interpretable basis decomposition (IBD) explains image classification model by showing relevant concepts that are human-interpretable [13]. In particular, concept activation vectors (CAV) are discussed by Kim et al. [12] as a framework to interpret latent representations in DNNs. This technique has been shown to be implemented by using supervised approaches where data with human-annotated concepts is available [12], or by unsupervised techniques (*i.e.* clustering) to retrieve relevant concepts directly from the training data [35].

III. Methodology

Our approach simplifies and visualises the otherwise complex representation of a latent space of any layer of a DNN. We factorize a desired layers' activation matrix to find k prototypes and their respective weights for each input instance. Using this post-hoc analysis protocol we probe an existing model and explain its reasoning process. We design our approach to be model and data agnostic by being able to work with a variety of DNN architectures for image, time-series, and text data analysis.

More specifically, as illustrated in Figure 1, we design ProtoFac to build a *surrogate model* to explain the original DNN's activation matrix at any user-specified layer l, which we denote as A^l. Assuming the latent representation at layer l is a fixed length vector with m dimensions and the total number of input instances is n, A^l will be a $n \times m$ matrix where each row $\boldsymbol{a}_i^l \in \mathbb{R}^m$ represents the latent activation of input instance $\boldsymbol{x}_i$ at layer l. ProtoFac decomposes A^l to obtain $A^l_{n\times m} \approx W_{n\times k} \cdot H_{k\times m}$, where k is the number of prototypes, a hyperparameter that needs to be specified. Each row $\boldsymbol{h}_j \in \mathbb{R}^m$ in $H_{k\times m}$ is a *prototype vector* and each row $\boldsymbol{w}_i \in \mathbb{R}^k$ in $W_{n\times k}$ is a *weight vector* to combine the k prototypes and recover the original activation vector $\boldsymbol{a}_i^l$ of $\boldsymbol{x}_i$. For the prototype vectors $\boldsymbol{h}_j$ $(0 \leq j < k)$ to be interpretable, in ProtoFac we constrain them to be the latent representations of realistic data samples or segments of data samples at layer l, *e.g.*, image patches, shapelets (*i.e.* segments in time-series) or n-grams in text data.

In Figure 1, $f^{l-}(\cdot)$ represents the downstream part in the original network after layer l and $f^l(\cdot)$ represents the upstream part that takes any input $\boldsymbol{x}_i$ and output the latent representation $\boldsymbol{a}_i^l = f^l(\boldsymbol{x}_i)$ at layer l. Using the original latent representation at layer l, the prediction for $\boldsymbol{x}_i$ is $\hat{y}_i$, which we refer to as the *oracle prediction*. The surrogate model uses the recovered activation $W \times H$ as input to the downstream layers

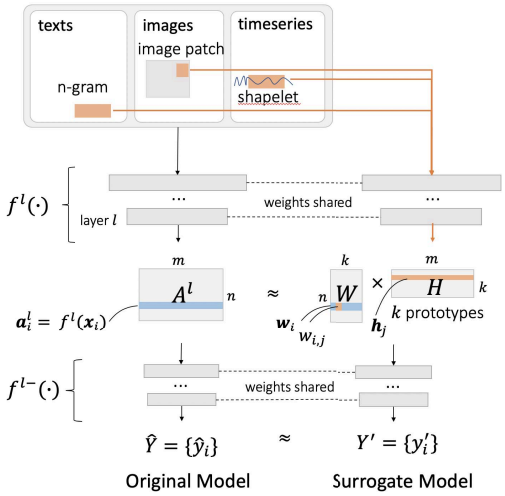

Fig. 1. ProtoFac uses a *surrogate model* that replaces the activation matrix $\boldsymbol{A^l}$ at any selected layer $\boldsymbol{l}$ in a neural network with weighted combinations of prototypes (*i.e.* $\boldsymbol{W} \times \boldsymbol{H}$). To *authentically* reflect the model operation the goal is to reconstruct the activation matrix with minimum uninterpreted residuals (*i.e.* $\|\boldsymbol{A^l} - \boldsymbol{W} \times \boldsymbol{H}\|_F$) and mimic the original models' prediction as much as possible. For better *interpretability*, we constrain the prototype vectors $\boldsymbol{h}_j$ in $\boldsymbol{H}$ to be the latent representations of realistic data samples or segments of data samples at layer l.

after l to obtain a new set of predictions for $\{\boldsymbol{x}_i\}$ which should highly resemble the original model's oracle predictions.

A. Optimization Objective

We formulate the optimization objective based on the desiderata listed in section I for post-hoc explanation of DNNs.

Authenticity. ProtoFac replaces the original model's activation matrix with the recovered activation matrix obtained through the weighted combination of prototype vectors and feeds it to the downstream network. We anticipate using this step it should produce similar prediction compared to the original network. To faithfully reflect the original model's behaviour, we define the following two loss terms:

- *Frobenius norm* of the factorization residual:

$$L_r(W,H)|_{X,f,l} = \frac{1}{n}||R||_F = \frac{1}{n}||A^l - W \times H||_F \quad (1)$$

where $X = \{\boldsymbol{x}_i\}, 0 \leq i < n$ represents all the input instances, f is the trained oracle model and l is the selected factorization layer. The goal is to minimize uninterpreted residuals if we replace the original activation matrix with the weighted combination of prototypes at layer l.

- *Cross entropy loss* comparing oracle model's and the interpretation surrogate's predictions, using binary classification as an example:

$$L_{ce}(W,H) = -\frac{1}{n}\sum_{0\leq i<n} \hat{y}_i \log(p'(\hat{y}_i)) + (1-\hat{y}_i)\log(1-p'(\hat{y}_i)) \quad (2)$$

where $\hat{y}_i$ is the oracle prediction on the input instance $\boldsymbol{x}_i$, and $p'(\hat{y}_i)$ is the surrogate model's predicted probability on

the oracle label, obtained by feeding reconstructed activation down through $f^{l-}(\cdot)$.

Non-negativity. We desire to find matrix W with only non-negative entries to allow only additive combinations of prototypes. We also constrain that each row in W to be summed to 1.0 such that the weights of the prototypes can be directly compared among different input instances.

Sparsity and concentration. To ensure that users are not overwhelmed by the shown prototypes, we seek to find less but good prototypes that can reconstruct the activation matrix precisely. To encourage that the distribution of the weight to be concentrated at only a few prototypes for each input, we add a *concentration* loss term:

$$L_c(W) = \frac{1}{n} \sum_{0 \le i < n} \min_{0 \le j < k} \|\boldsymbol{w}_i - \boldsymbol{e}_j\|^2 \tag{3}$$

where $\boldsymbol{e}_j$s are standard basis vectors with length k. Only the jth entry in $\boldsymbol{e}_j$ is equal to 1.0 and all the others are equal to zero. The loss function encourages the weights to concentrate on any one prototype. Notice that this is a soft-constraint and does not enforce a strict clustering boundary as k-means does.

Full objective. We combine the above discussed loss terms and constraints together to form the following optimization objective:

$$\begin{aligned} Loss(W,H)|_{X,f,l} =& \lambda_{ce} L_{ce}(W,H)|_{X,f,l} + \lambda_r L_r(W,H)|_{X,f,l} \\ &+ \lambda_c L_c(W) \end{aligned} \tag{4}$$

where $W \in \mathbb{R}^{n \times k}$, $H \in \mathbb{R}^{k \times m}$, $W \ge 0$, $H \ge 0$ and $\sum_{0 \le j < k} w_{i,j} = 1.0$.

B. The ProtoFac Algorithm

With the additional loss terms in the optimization objective matrix factorization techniques *e.g.* alternating least squares (ALS) is no longer sufficient. The optimization objective is not convex with respect to W or H due to the addition of the authenticity term involving the downstream layers $f^{l-}(\cdot)$ in the deep neural network. Therefore we propose an algorithm using stochastic gradient descent (SGD) with mini-batch to obtain the prototypes and their respective weights.

The ProtoFac algorithm is shown in detail in Algorithm 1. It first collects the activation matrix A^l and the oracle predictions $Y = \{\hat{y}_i\} (0 \le i < n)$ by feeding the training data $X = \{\boldsymbol{x}_i\}$ into the original DNN (`line 1-2`). The activation matrix is constructed by flattening the latent activation of each input at layer l and concatenate them to form an $n \times m$ matrix. After that, a set of candidate prototypes are generated by first randomly sampling a subset of X and then applying $g(\cdot)$ to each sample $x_i \in sampler(X)$ to generate a set of candidate prototypes. $g(\cdot)$ varies for different types of data but generally it can be implemented by applying a sliding window over *e.g.* image or time-series data to obtain a set of image patches or shapelets respectively. We collect all the candidate prototypes $P = \cup_{\boldsymbol{x}_i \in sampler(X)} g(\boldsymbol{x_i})$ as well as their latent representations at layer l, which are collectively denoted as A^l_P `line 3-4` . For DNNs that accept varying lengths inputs, the candidate prototypes are directly fed into the network to obtain the latent representation. For DNNs with fixed size inputs we simply mask the data outside the region covered by the moving window.

Input: pretrained model f, selected layer l, training data $X = \{\boldsymbol{x}_i\}$, candidate prototype generator $g(\boldsymbol{x}_i)$
Parameters: number of prototypes k, hyperparameters (λs)
Output: prototype vector H, weight matrix W

```
/* Obtain activation matrix and
   oracle labels                                   */
```
1 $A^l = [\boldsymbol{a}_i]$, $\boldsymbol{a}_i = f^l(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in X$;
2 $\hat{Y} = \{\hat{y}_i = f(\boldsymbol{x}_i)\}$, $\boldsymbol{x}_i \in X$;
```
/* Obtain candidate prototypes and
   their latent activations                        */
```
3 $P = \cup_{\boldsymbol{x}_i \in sampler(X)} g(\boldsymbol{x_i})$;
4 $A^l_P = [\boldsymbol{a}_p]$, $\boldsymbol{a}_p = f^l(p)$, $p \in P$;
```
/* Freeze up and downstream network
   in oracle model                                 */
```
5 freeze_parameter(θ) for θ in $f^l(\cdot)$ and $f^{l-}(\cdot)$;
6 **for** *epoch* $\in [1, n\_epochs]$ **do**
7 **for** *batch* $\in$ *batch_generator(*A^l*.rows)* **do**
8 $batch\_loss = loss(W[batch.rows], H)$;
9 update $W[batch.rows]$ and H with gradient descent;
10 **end**
11 **if** *mod(epoch, projection_interval)* $= 0$ **then**
```
/* project to candidate
   prototypes                                      */
```
12 $H = [\boldsymbol{h}_j]$ where $\boldsymbol{h}_j = f^l(p_j)$, $p_j = argmin_{p \in P} \|\boldsymbol{h}_j - f^l(p)\|^2$;
```
/* freeze H and update W                           */
```
13 **for** *epoch'* $\in [1, n\_epochs']$ **do**
14 **for** *batch* $\in$ *batch_generator(*A^l*.rows)* **do**
15 $batch\_loss = loss(W[batch.rows], H)$;
16 update $W[batch.rows]$ with gradient descent;
17 **end**
18 **end**
19 **end**

Algorithm 1: The ProtoFac algorithm.

Before the training starts we freeze the parameters in both the upstream and downstream layers (`line 5`) since we want to keep the oracle model intact. During training, W and H are initialized with random weights and updated through SGD (Adam [37] optimizer is used in the experiments presented in this paper). We combine rows in A^l to form training batches (`line 7`) to handle large scale data. When iterating through each batch the corresponding rows in W and the entire H will be updated through gradient descent (`line 8-9`). For every few epochs and also after the last epoch we perform prototype projection (`line 11-18`) which first assigns the

prototype vectors $\boldsymbol{h}_j$ obtained through gradient descent to their nearest neighbors in P in euclidean distance (`line 12`). The respective image patches, shapelets and n-grams are stored accordingly to generate user-friendly explanations along with the weights. After projection the algorithm freezes the prototype vectors and updates the weights again through SGD (`line 13-18`) to obtain an optimal factorization. The training process stops when the accuracy of the surrogate model with respect to the oracle prediction no longer improves.

With ProtoFac described in Algorithm 1 we can obtain a set of prototypes and their corresponding weights for a training set. To evaluate the applicability of the identified prototypes to unseen data we can use a similar algorithm except that now the prototype matrix H need to be freezed and the algorithm no longer performs prototype projection. A new W matrix is obtained for the unseen data however the same prototypes are used as for the training set.

IV. Experimental Evaluation

In this section we report experimental results on a variety of DNNs for different ML tasks. All the experiments are conducted on publicly available datasets including image, time-series, and text data. We also conduct various ablation studies to examine how different hyperparameter settings, and the selection of different factorization layers in a model affects the surrogate model's accuracy. We also conduct user study to evaluate human interpretability of the factorized prototypes.

We implement the DNN models and ProtoFac using PyTorch[1]. We utilize trained oracle models and save their internal parameters. The latent activations at the selected layer are collected through implementing a hook function in PyTorch and running the training samples through the network. In the same way we collect the latent activations of the prototype candidates. When training the surrogate model all the downstream layer parameters in the oracle model are freezed.

A. Case Study: Interpret Image Classifiers: VGG and ResNet

We apply ProtoFac to analyze two models for image classification: VGG19 (+batchnorm) [38] and ResNet50 [39]. Both models are trained on the CIFAR-10 dataset [40], which contains 60000 colored images evenly distributed in 10 classes. Each image has a resolution of 32×32. The models [2] have more than 94% validation accuracy.

We select two layers each from VGG19 and ResNet50 for the experiment (Table I). The feature map of the selected layer is flattened to collect the activation matrix. In the surrogate model, after obtaining the reconstructed activation we also reshape it accordingly in order to send it to the downstream network. For more details about the original models' architectures and the layers selected for extracting prototypes please refer to Appendix VI-B. The prototype candidates are image patches generated from the training samples with a moving window of size 16×16 and a stride of 4. Therefore for each image 5×5 image patches are created. We also experimented with image patches of size 4× 4, 8×8, 16×16 respectively and found 16×16 gives the best results in terms of the authenticity with respect to the original model. To limit the number of patches, we uniformly sampled 20% images for each class. For all the experiments with different layer and model combinations, we train the surrogate model using batch size of 64 and a learning rate of 0.005. In total for each experiment we run 40 training epochs with a projection frequency of 5 and report the best result (in terms of accuracy wrp. the oracle model) obtained in the training process. More detailed hyperparameter settings can be found in Appendix VI-A.

Table I summarizes the experimental results. The result shows that the surrogate model can achieve high fidelity to the original model - the accuracy of the surrogate models with respect to the oracle models' predictions (Acc. (vs. oracle) in Table I) remains high around 99% with appropriate setting of prototype number k. Correspondingly, the surrogate models also has similar accuracy as the oracle model with respect to ground truth labels (Acc. (vs. groundtruth) in Table I). The Frobenius losses (F-loss (ProtoFac) in Table I) remain reasonably close and sometimes is even lower compared to the one obtained through a classic non-negative matrix factorization algorithm [20], [21] (F-loss (NMF)) [3]. Comparing the layer maxpool3 and maxpool5 results for VGG19 with equal k, we also observe that by factorizing the layer closer to the output the algorithm can achieve higher fidelity to the oracle model, which is not too surprising. In Figure 5 we conducted more extensive experiment to analyze how the selection of different k and layers in the original model would affect the performance of the surrogate model.

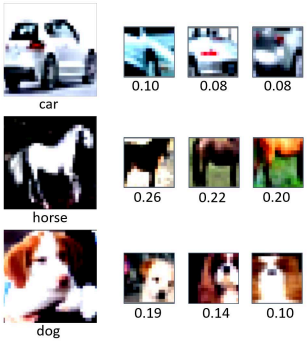

Fig. 2. Example image patches and the highest weighted prototypes. The first row shows the prototypes associated with a car image: one prototype contains the wheel and another contains the red light which could be associated with the tail lamp. On the second row the horse is recognized by its body shape as the highest weighted prototypes all describe body shapes.

Figure 2 shows example prototypes along with their weights from the factorization results to explain the original model's prediction. The result shown in the figure is obtained by factorizing the maxpool3 (Figure 9) layer in VGG19. It clearly shows that some predictions are performed by using a parts-

[1] https://pytorch.org/

[2] We use the pretrained models from https://github.com/huyvnphan/PyTorch-CIFAR10

[3] We use the implementation at http://nimfa.biolab.si/ and set the rank to k.

TABLE I
EXPERIMENTAL RESULTS ON VGG AND RESNET FOR IMAGE CLASSIFICATION TASKS.

Dataset	Model	Acc.(valid)	Factorized Layer	k	Acc.(vs. oracle)	Acc.(vs. groundtruth)	F-loss(ProtoFac)	F-loss(NMF)
CIFAR-10	VGG19	94.25	maxpool3	60	96.10%	90.65%	0.0006	0.0009
			maxpool3	120	98.45%	92.80%	0.0006	0.0009
			maxpool5	60	100.00%	93.60%	0.0014	0.0243
	ResNet50	94.38	bottleneck14	60	98.35%	94.15%	0.0006	0.0056
			bottleneck14	120	99.15%	94.30%	0.0007	0.0056
			bottleneck16	60	99.65%	94.35%	0.0007	0.0197

TABLE II
EXPERIMENTAL RESULTS ON RESNET-1D FOR TIME-SERIES CLASSIFICATION TASK ON THE MIT-BIH DATA.

Dataset	Model	Acc.(valid)	Factorized Layer	k	Acc.(vs. oracle)	Acc.(vs. groundtruth)	F-loss(ProtoFac)	F-loss(NMF)
MIT-BIH	ResNet-1D	98.23	block1	60	95.10%	81.21%	1.812	1.9113
			block2	50	97.63%	95.94%	1.072	1.123
			block3	50	98.21%	97.27%	0.873	0.943
			fc	50	100.00%	98.34%	0.0402	0.0654

based representation: on the first row the image is classified as a car since it is related to prototypes containing the wheel and the red taillight and the car back individually. Figure 3 shows some example prototypes from different classes and the image samples with the highest weights on those prototypes.

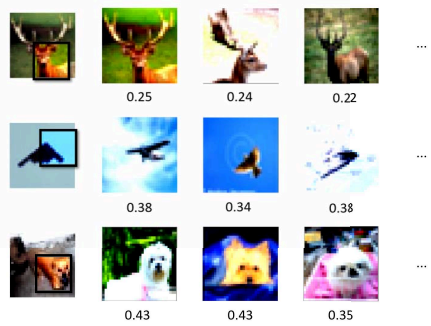

Fig. 3. Example prototypes (highlighted in their source images) and images with heavy weights on those prototypes. On the second row both birds and airplanes are matched to the same prototype for their similar wing shapes.

B. Case Study: Interpret Time Series Classifiers for ECG Data

Electrocardiogram (ECG) records are widely utilised by medical practitioners to monitor patients' cardiovascular health and perform diagnosis. Since manual analysis of ECG signals is both time-consuming and error-prone, recently a number of studies explore using machine learning to automatically perform anomaly detection or classification on ECG signals. Among the ML models DNNs is one of the most widely used.

We test our technique on a DNN model to classify ECG signals, using the MIT-BIH Arrhythmia ECG Databases [41], [42] with labeled records. The dataset contains ECG recordings from 47 subjects each recorded at a sampling rate of 360Hz. We use preprocessed data from [43] where each segment corresponds to a heatbeat. In accordance with Association for the Advancement of Medical Instrumentation (AAMI) EC57 standard [44], each of the segments are annotated with one of the 5 labels: Normal (N), Supraventricular Ectopic Beat (SVEB), Ventricular Ectopic Beat (VEB), Fusion Beat (F), and Unknown Beat (Q). Furthermore the data is divided into training and validation set with 87k samples and 21k samples, respectively. Since the ECG data is a uni-variate time series, we utilised a 1D CNN model. (architecture diagram in Appendix VI-B). We train the CNN-1D model with convolutional kernels of size $4, 8, 16, 32, 64$ and 128 channels each, a max pooling (over time) layer, and 2 fully connected layers following that. The model is trained with batch size of 4096. With 120 epochs we obtain an original model with 99.37% and 98.11% training and validation accuracy (Table III).

TABLE III
EXPERIMENTAL RESULTS ON CNN-1D MODEL FOR ECG TIME-SERIES CLASSIFICATION.

Dataset	Model	Acc. (valid)	Factor. Layer	k	Acc. (v. oracle)
MIT-BIH	CNN	98.11 %	fc1	50	99.76 %
			fc2	50	100.00 %

cont.	Acc. (v. groundtruth)	F-loss (ProtoFac)	F-loss (NMF)
	97.76 %	0.0132	0.0231
	98.09 %	0.0651	0.0320

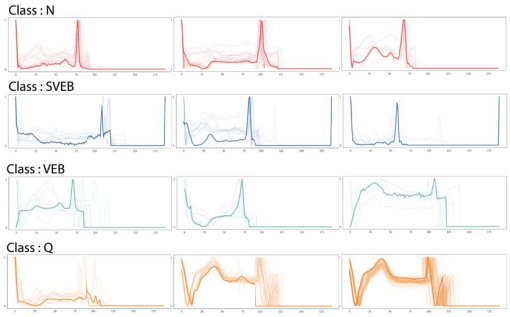

Fig. 4. Recovered prototypes for ECG data. Each class is represented with a separate color. The solid line is the prototype while the transparent lines are inputs with the highest weight on the corresponding prototypes.

For the experiments on ECG data, we use complete heatbeat sequences as candidate prototypes and do not apply moving window on top of it to extract time series segments

as prototypes. The reason is that the original sequences only contain individual heatbeats and further dividing them could hurt interpretability. We train the surrogate model using $k = 50$ with 120 epochs and a projection frequency of 30. (for detailed training hyperparameter settings please refer to Appendix VI-A). We factorize the output from the two layers just before fc1 and fc2 and find that our surrogate model is able to obtain high fidelity with respect to the original model (Table III Acc. (vs. oracle)) at both layers. The activation matrix is also reconstructed with reasonable Frobenious losses (Table III F-loss (ProtoFac)) when compared to traditional NMF technique (Table III F-loss (NMF)).

Our analysis using visualizations (Figure 4) show that these prototypes are good representatives of the ECG data samples. We also categorize the prototypes by class labels to analyze if the prototypes capture some distinctive features of that class. We find that the prototypes that correspond to class label SVEB and class label VEB have more irregular rhythms compared to the Normal Beats (N) with varying positions of peaks. Prototypes associated the class label Unknown Beat (Q) on the other hand shows a lot of diversity and variation (Figure 4).

C. Experiments to verify our matrix factorization approach:

To validate our technique on the MIT-BIH ECG timeseries dataset we also deployed ProtoFac on a ResNet-1D model as introduced in [45]. The architecture for this model included 3 'blocks' with kernel sizes $[7, 5, 3]$, and channel sizes of each as $[64, 128, 128]$. Each 'block' is composed of 3 1D-convolution layers (each followed by a batch normalization function). Before making prediction, we connect the output from all the 'block' layers to a fully connected layer. To guard for overfitting, we use a dropout rate of 0.2. The model is trained with batch size of 512, learning rate of 0.007, and 80 epochs to get the best ground truth accuracy of 98.34% on the validation set. In ResNet-1D we tested ProtoFac's effectiveness by factorizing the layers 'block1', 'block2', 'block3', and 'fully connected', one at a time (refer Table II). While we factorized these layers' we froze the parameters in the up and downstream layers of this model in order to preserve the oracle model. As we train the surrogate model, we initialize W and H with random weights and then train the weights using SGD (with Adam as the optimization algorithm). W and H matrices are updated per iteration in the gradient descent's training process; after finishing an epoch, ProtoFacretrieves 'k' prototypes. We conducted the following experiments on this network to further verify the effectiveness of ProtoFac.

Comparing with other matrix factorization methods: We compared the accuracy metric of our surrogate model when the activation matrix was factorized using ProtoFac vs. when factorized with traditional non-negative matrix factorization techniques. We used the NIMFA python library's [46] NMF method and assigned the 'explained variance' as the objective function and 'euclidean' as the update metric as input parameters. We found that using ProtoFac the ground truth accuracy of the surrogate model was 98.34% on the ECG Dataset, while using NMF method from NIMFA, the accuracy was 96.65% (factorization layer was 'fully connected' layer). The ground truth accuracy results were 95.94% and 95.02% for ProtoFac and NIMFA respectively when the layer 'block2' was factorized. The Frobenious loss compare to traditional NMF method as shown in Table II shows that our method also consistently performs better to recover the original activation matrix. This proves that our matrix factorization approach performed comparably well with other factorization methods. However, in ProtoFac while we factorized the activation matrix we also recovered prototypes explaining the original model with semantically meaningful image patches or shapelets.

Activation Matrix reconstruction: Next, we sought to verify the effectiveness of ProtoFac to accurately reconstruct the original activation matrix even if there are any missing values in it. To test this, we programmatically replaced 20% of the original values from the activation matrix with null values (represented by 0). Then using ProtoFac we factorized this activation matrix (with part null values). Our results show that when the 'fully connected' layer was factorized the ground truth accuracy dropped by only 3.42%, thus proving that our approach of matrix factorization very closely reconstructs the original matrix even if there are missing values in it.

D. Ablation Studies

Effect of the number of prototypes k: We are curious to study how the number of prototypes k impacts the accuracy of the surrogate model. We begin the experiment with a low value of $k = 3$ and then gradually increase it to study how the surrogate model's accuracy change with respect to both the oracle model's prediction and the ground truth labels. The experiments are conducted on both CNN-1D for ECG data analysis and VGG19 for image classification. Two layers are selected from each model for the experiment, same as the ones in Table I and Table III. All the experimental results are obtained on a held-out validation dataset.

Figure 5 summarizes the results. For both models we observe that as we increase k the accuracy of the surrogate model gradually increased and then flattened out for larger k's. The accuracy with respect to the oracle model predictions saturates near 100% and the accuracy with respect to the ground truth labels saturates at the oracle model's validation accuracy. The result shows that with sufficient number of prototypes the surrogate model is able to accurately approximate the original model's output and adding more prototypes after the model saturates has diminishing marginal utility. The curve can also be used to select an appropriate number of prototypes. One approach that worked very well for us, was to start with a low value of k and then increase it until we do not observe any significant change in the model's accuracy. In addition, one should consider that having a surrogate model with a high number of prototypes may render the model less interpretable by adding undesirable prototypes as noise.

Effect of the selected layer for prototype factorization: Figure 5 also shows how the behaviour of the surrogate model changes as different layers from DNNs are selected for prototype factorization. For both CNN-1D and VGG19, we

observe that as the selected layer move closer to output (fc2 in CNN-1D and maxpool5 in VGG19), the surrogate model's performance saturates much faster as k is increased. The reason is that the latter layers generate latent representations that can be more easily separated for prediction.

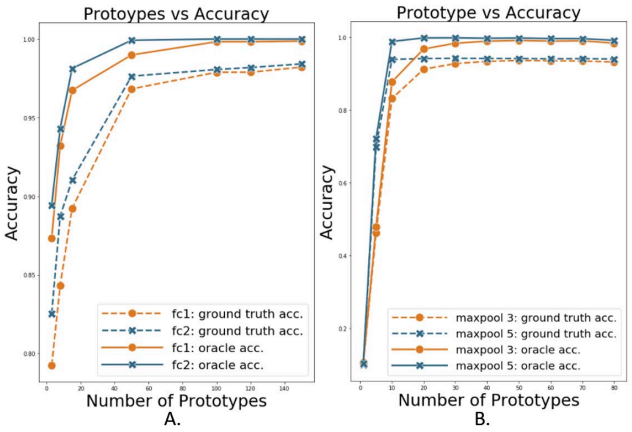

Fig. 5. Plot of surrogate model's accuracy (v. ground truth and oracle) in relation to the number of prototypes k. **A**. accuracy vs. k for the CNN-1D for ECG classification. Note the data is from the two fully connected layers in the CNN model. fc2 is the penultimate layer. **B**. accuracy vs. k for CIFAR-10 on VGG19 maxpool3 and maxpool5 layers (Figure 9).

E. Crowd-sourced evaluation of Interpretability

Interpretation of a model by non-experts are often driven by subjective aspects. Thus to evaluate effectiveness of our method in helping users interpret models with the aid of prototypes, we conduct a quantitative evaluation of ProtoFac with human subjects. Through this experiment we seek to answer how interpretable and understandable are the prototypes in explaining the prediction of a trained DNN model. For the evaluation, we use the VGG19 model trained on CIFAR-10 image classification data (10 class labels) with 60 prototypes extracted from maxpool3. To collect user feedback on the model interpretation we recruit human participants on Amazon Mechanical Turk (MTurk) who are non-experts in machine learning. We ask users to fill a survey questionnaire with 20 questions each for image and text data.

Experiment Settings and Results (VGG): We generated a set of 20 questions where each question contains an image (we sampled two images from each class in CIFAR-10) with a class label and a set of six candidate prototypes as potential explanations to the prediction of the image (see Figure 6). Users were asked the following question: "Which of the following options do you think can be used to explain the image (on the left) and its caption (label)?" If none of the shown prototypes explain the image and its label, then users can choose the last option "None of them". Out of the 6 candidate prototypes 2 were prototypes selected by the ProtoFac to explain the prediction, 2 were other prototypes, and 2 were randomly chosen image patches . Through MTurk we collected 58 responses and removed 6 of them for missing entries. From the remaining 52 responses we analysed the data to find that on average the users' selections align with the algorithm selections for 16.314 (SD = 2.37) out of the 20 input images (we consider they are aligned if the user chooses any of the two prototypes). From this result we can conclude that most of the prototypes generated by our surrogate model are human understandable explanations of the predictions. Figure 7 analyze the distribution of the average alignment score (percentage of aligned responses) for different classes and the distribution of the average alignment score for different experiment subjects.

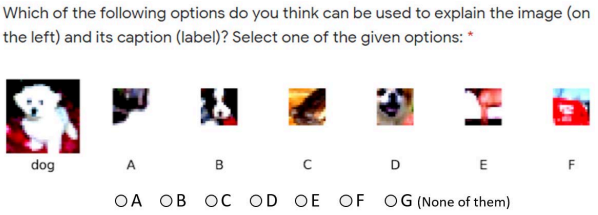

Fig. 6. Example question used in our user study with the image data.

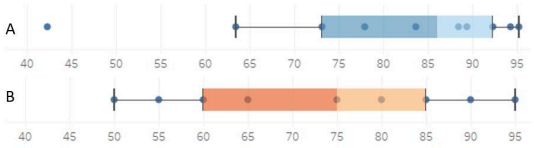

Fig. 7. Box-plots show **A.** the distributions of the average alignment score (percentage of aligned responses) for different classes and **B.** for different users. The result is for VGG model on CIFAR-10.

V. Conclusion and Future Work

We present a post-hoc, model-agnostic interpretation method for general DNNs. The proposed matrix factorization algorithm named ProtoFac decomposes the latent activation in any selected layer in a DNN into a set of prototypes with corresponding weights. We formulate a novel optimization objective for ProtoFac considering the various desiderata to obtain post-hoc interpretations of ML models including *authenticity*, *interpretability*, and *simplicity* and propose the corresponding optimization procedure. Through experiments on a variety of DNN architectures for different ML tasks such as time series classification on ECG data and image classification, we demonstrate that our algorithm is able to find a set of meaningful prototypes to explain the model's behaviour globally while remaining truthful to reflect the underlying model's operations. We also conducted a large scale user study on Amazon Mechanical Turk to evaluate the human interpretability of the extracted prototypes. The results demonstrate that the algorithm is able to extract prototypes that can be easily understood and align well with human intuition and common sense. While the first step is promising, continued effort and further research is needed to scale the solution for larger datasets, more complex models, and for a diverse set of ML tasks.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 1097–1105.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[3] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, 2017.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[5] A. Holzinger, "Interactive machine learning for health informatics: when do we need the human-in-the-loop?" *Brain Informatics*, vol. 3, no. 2, pp. 119–131, Jun 2016. [Online]. Available: https://doi.org/10.1007/s40708-016-0042-6

[6] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[7] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," 06 2017.

[8] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 3319–3328. [Online]. Available: http://proceedings.mlr.press/v70/sundararajan17a.html

[9] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," *CoRR*, vol. abs/1602.04938, 2016. [Online]. Available: http://arxiv.org/abs/1602.04938

[11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901

[12] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. Sayres, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)," 2017.

[13] B. Zhou, Y. Sun, D. Bau, and A. Torralba, "Interpretable basis decomposition for visual explanation," in *ECCV*, 2018.

[14] J. L. Kolodner, "An introduction to case-based reasoning," *Artificial Intelligence Review*, vol. 6, no. 1, pp. 3–34, Mar 1992. [Online]. Available: https://doi.org/10.1007/BF00155578

[15] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *Proceedings of AAAI*, 2018.

[16] Y. Ming, P. Xu, H. Qu, and L. Ren, "Interpretable and steerable sequence learning via prototypes," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '19. New York, NY, USA: ACM, 2019.

[17] A. B. J. S. C. R. Chaofan Chen, Oscar Li, "This looks like that: Deep learning for interpretable image recognition," in *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2019.

[18] P. Hase, C. Chen, O. Li, and C. Rudin, "Interpretable image recognition with hierarchical prototypes," *CoRR*, vol. abs/1906.10651, 2019. [Online]. Available: http://arxiv.org/abs/1906.10651

[19] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, no. 1, p. 68–77, Dec. 2019. [Online]. Available: https://doi.org/10.1145/3359786

[20] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001, pp. 556–562.

[21] D. L. Daniel and H. S. Seung, "Learning the parts of objects by nonnegative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[22] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model," 2015.

[23] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1721–1730.

[24] B. Ustun and C. Rudin, "Supersparse linear integer models for optimized medical scoring systems," *Machine Learning*, vol. 102, pp. 349–391, 2015.

[25] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. PMLR, 07–09 Jul 2015, pp. 2048–2057.

[26] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," in *NIPS*, 2017.

[27] D. Erhan, Y. Bengio, A. C. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," 2009.

[28] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, and O. D. Suárez, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," in *PloS one*, 2015.

[29] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," *ArXiv*, vol. abs/1704.02685, 2017.

[30] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards better understanding of gradient-based attribution methods for deep neural networks," in *ICLR*, 2018.

[31] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," 2017.

[32] A. Ghorbani, A. Abid, and J. Y. Zou, "Interpretation of neural networks is fragile," in *AAAI*, 2017.

[33] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim, "The (un)reliability of saliency methods," in *Explainable AI*, 2018.

[34] J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *NeurIPS*, 2018.

[35] A. Ghorbani, J. Wexler, and B. Kim, "Automating interpretability: Discovering and testing visual concepts learned by neural networks," *ArXiv*, vol. abs/1902.03129, 2019.

[36] C.-K. Yeh, B. Kim, S. O. Arik, C.-L. Li, P. Ravikumar, and T. Pfister, "On concept-based explanations in deep neural networks," 2019.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[40] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[41] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," pp. 215–220, 2003.

[42] E. M. Dataset, https://www.physionet.org/content/mitdb/1.0.0/, accessed: 2020-02-05.

[43] M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "Ecg heartbeat classification: A deep transferable representation," *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/ICHI.2018.00092

[44] ANSI/AAMI, "Testing and reporting performance results of cardiac rhythm and st segment measurement algorithms," *Standard. American National Standards Institute, Inc. (ANSI), Association for the Advancement of Medical Instrumentation (AAMI)*, 2008.

[45] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International*

joint conference on neural networks (IJCNN). IEEE, 2017, pp. 1578–1585.

[46] M. Zitnik and B. Zupan, "Nimfa: A python library for nonnegative matrix factorization," *Journal of Machine Learning Research*, vol. 13, pp. 849–853, 2012.

VI. Experiment Details

A. Hyperparameter Settings

In Table I, we set $\lambda_{ce} = 1.5$, $\lambda_r = 50.0$, and $\lambda_c = 10.0$. Other training configs are: $n\_epochs = 50$, $batch\_size = 64$, $projection\_interval = 10$, $learning\_rate = 0.005$, $n\_epochs' = 20$, and $learning\_rate\_weight\_updates = 0.005$.

In Table III, for the experiment on the CNN model for electrocadiodiagram (ECG) classification, we set $\lambda_{ce} = 30.0$, $\lambda_r = 15.0$ and $\lambda_c = 1.0$. Other training configs are: $k = 50$, $n\_epochs = 120$, $batch\_size = 4096$, $projection\_interval = 30$, $learning\_rate = 0.09$, $n\_epochs' = 20$, and $learning\_rate\_weight\_updates = 0.005$.

In Figure 5, for the experiment on VGG19, we set $\lambda_{ce} = 1.5$, $\lambda_r = 50.0$ and $\lambda_c = 10.0$. Other training configs are: $n\_epochs = 39$, $batch\_size = 64$, $projection\_interval = 5$, $learning\_rate = 0.005$, $n\_epochs' = 20$, and $learning\_rate\_weight\_updates = 0.005$.

B. Model Architectures

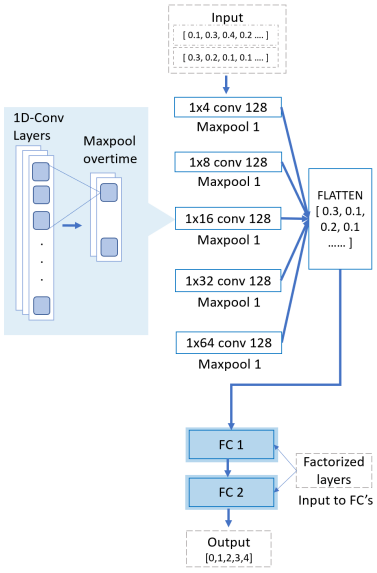

Fig. 8. CNN-1D model architecture for ECG data.

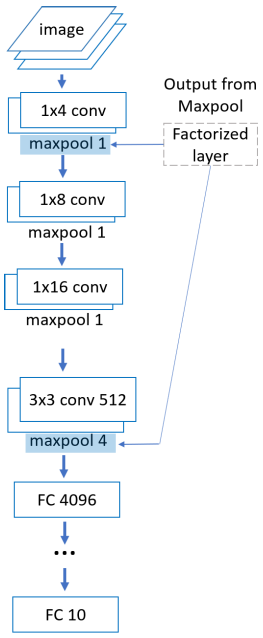

Fig. 9. VGG19 model architecture for CIFAR-10.

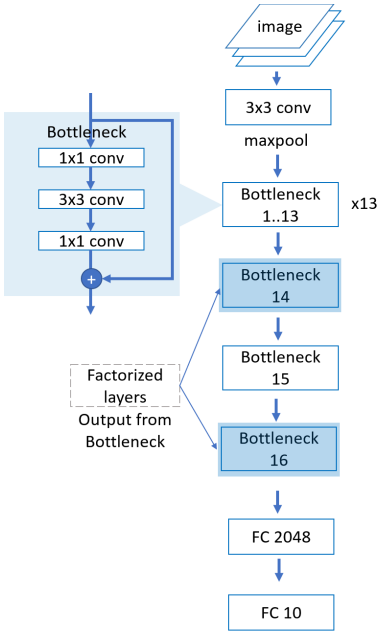

Fig. 10. ResNet50 model architecture for CIFAR-10.