Explaining Deep Learning Models – A Bayesian Non-parametric Approach

Wenbo Guo
The Pennsylvania State University
wzg13@ist.psu.edu

Sui Huang
Netflix Inc.
shuang@netflix.com

Yunzhe Tao
Columbia University
y.tao@columbia.edu

Xinyu Xing
The Pennsylvania State University
xxing@ist.psu.edu

Lin Lin
The Pennsylvania State University
llin@psu.edu

Abstract

Understanding and interpreting how machine learning (ML) models make decisions have been a big challenge. While recent research has proposed various technical approaches to provide some clues as to how an ML model makes individual predictions, they cannot provide users with an ability to inspect a model as a complete entity. In this work, we propose a novel technical approach that augments a Bayesian non-parametric regression mixture model with multiple elastic nets. Using the enhanced mixture model, we can extract generalizable insights for a target model through a global approximation. To demonstrate the utility of our approach, we evaluate it on different ML models in the context of image recognition. The empirical results indicate that our proposed approach not only outperforms the state-of-the-art techniques in explaining individual decisions but also provides users with an ability to discover the vulnerabilities of the target ML models.

1 Introduction

When comparing with relatively simple learning techniques such as decision trees and K-nearest neighbors, it is well acknowledged that complex learning models – particularly, deep neural networks (DNN) – usually demonstrate superior performance in classification and prediction. However, they are almost completely opaque, even to the engineers that build them [20]. Presumably as such, they have not yet been widely adopted in critical problem domains, such as diagnosing deadly diseases [13] and making million-dollar trading decisions [14].

To address this problem, prior research proposes to derive an interpretable explanation for the output of a DNN. With that, people could understand, trust and effectively manage a deep learning model. From a technical prospective, this can be interpreted as pinpointing the most important features in the input of a deep learning model. In the past, the techniques designed and developed primarily focus on two kinds of methods – (1) *whitebox explanation* that derives interpretation for a deep learning model through forward or backward propagation approach [26, 36], and (2) *blackbox explanation* that infers explanations for individual decisions through local approximation [21, 23]. While both demonstrate a great potential to help users interpret an individual decision, they lack an ability to extract insights from the target ML model that could be generalized to future cases. In other words, existing methods could not shed lights on the general sensitivity level of a target model to specific input dimensions and hence fall short in foreseeing when prediction errors might occur for future cases.

In this work, we propose a new technical approach that not only explains an individual decision but, more importantly, extracts generalizable insights from the target model. As we will show in

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

Section 4, we define such insights as the *general sensitivity level of a target model to specific input dimensions*. We demonstrate that model developers could use them to identify model strengths as well as model vulnerabilities. Technically, our approach introduces multiple elastic nets to a Bayesian non-parametric regression mixture model. Then, it utilizes this model to approximate a target model and thus derives its generalizable insight and explanation for its individual decision. The rationale behind this approach is as follows.

A Bayesian non-parametric regression mixture model can approximate arbitrary probability density with high accuracy [22]. As we will discuss in Section 3, with multiple elastic nets, we can augment a regression mixture model with an ability to extract patterns (generalizable insights) even from a learning model that takes as input data of different extent of correlations. Given the pattern, we could extrapolate input features that are critical to the overall performance of an ML model. This information can be used to facilitate one to scrutinize a model's overall strengths and weaknesses. Besides extracting generalizable insights, the proposed model can also provide users with more understandable and accountable explanations. We will demonstrate this characteristic in Section 4.

2 Related Work

Most of the works related to model interpretation lie in demystifying complicated ML models through *whitebox* and *blackbox* mechanisms. Here, we summarize these works and discuss their limitations. It should be noticed that we do not include those works that identify training samples that are most responsible for a given prediction (*e.g.*, [12, 15]) and those works that build a self-interpretable deep learning model [7, 33].

The whitebox mechanism augments a learning model with the ability to yield explanations for individual predictions. Generally, the techniques in this kind of mechanism follow two lines of approaches – ❶ occluding a fraction of a single input sample and identifying what portions of the features are important for classification [4, 6, 17, 36, 37], and ❷ computing the gradient of an output with respect to a given input sample and pinpointing what features are sensitive to the prediction of that sample [1, 8, 24, 25, 26, 29, 32]. While both can give users an explanation for a single decision that a learning model reach, they are not sufficient to provide a global understanding of a learning model, nor capable of exposing its strengths and weaknesses. In addition, they typically cannot be generally applied to explaining prediction outcomes of other ML models because most of the techniques following this mechanism are designed for a specific ML model and require altering that learning model.

The blackbox mechanism treats an ML model as a black box, and produces explanations by locally learning an interpretable model around a prediction. For example, `LIME` [23] and `SHAP` [21] are the same kind of explanation techniques that sample perturbed instances around a single data sample and fit a linear model to perform local explanations. Going beyond the explanation of a single prediction, they both can be extended to explain the model as a complete entity by selecting a small number of representative individual predictions and their explanations. However, explanations obtained through such approaches cannot describe the full mapping learned by an ML model. In this work, our proposed technique derives a generalizable insight directly from a target model, which provides us with the ability to unveil model weaknesses and strengths.

3 Technical Approach

3.1 Background

A Bayesian non-parametric regression mixture model (*i.e.*, mixture model for short) consists of multiple Gaussian distributions:

$$y_i|\mathbf{x}_i, \boldsymbol{\Theta} \sim \sum_{j=1}^{\infty} \pi_j N(y_i \mid \mathbf{x}_i\boldsymbol{\beta}_j, \sigma_j^2), \tag{1}$$

where $\boldsymbol{\Theta}$ denotes the parameter set, $\mathbf{x}_i \in \mathbb{R}^p$ is the i-th data sample of the sample feature matrix $\mathbf{X}^{\mathrm{T}} \in \mathbb{R}^{p\times n}$, and y_i is the corresponding prediction in $\mathbf{y} \in \mathbb{R}^n$, which is the predictions of n samples. $\pi_{1:\infty}$ are the probabilities tied to the distributions with the sum equal to 1, and $\boldsymbol{\beta}_{1:\infty}$ and $\sigma^2_{1:\infty}$ represent the parameters of regression models, with $\boldsymbol{\beta}_j \in \mathbb{R}^p$ and $\sigma_j^2 \in \mathbb{R}$.

In general, model (1) can be viewed as a combination of infinite number of regression models and be used to approximate any learning model with high accuracy. Given a learning model $g : \mathbb{R}^p \rightarrow \mathbb{R}$, we can therefore approximate $g(\cdot)$ with a mixture model using $\{\mathbf{X}, \mathbf{y}\}$, a set of data samples as well as their corresponding predictions obtained from model g, *i.e.*, $y_i = g(\mathbf{x}_i)$. For any data sample $\mathbf{x}_i$, we can then identify a regression model $\hat{y}_i = \mathbf{x}_i\boldsymbol{\beta}_j + \epsilon_i$, which best approximates the local decision boundary near $\mathbf{x}_i$[1].

Note that in this paper, we assume that a single mixture component is sufficient to approximate the local decision boundary around $\mathbf{x}_i$. Despite the assumption doesnot hold in some cases, the proposed model can be relaxed and extended to deal with these cases. More specifically, instead of directly assigning each instance to one mixture component, we can assign an instance at a mode level [10], (*i.e.*, assigning the instance to a combination of multiple mixture components). When explaining a single instance, we can linearly combine the corresponding regression coefficients in a mode.

Recent research [23] has demonstrated that such a linear regression model can be used for assessing how the feature space affects a decision by inspecting the weights (model coefficients) of the features present in the input. As a result, similar to prior research [23], we can take this linear regression model to pinpoint the important features and take them as an explanation for the corresponding individual decision.

In addition to model approximation and explanation mentioned above, another characteristic of a mixture model is that it can enable multiple training samples to share the same regression model and thus preserve only dominant patterns in data. With this, we can significantly reduce the amount of explanations derived from training data and utilize them as the generalizable insight of a target model.

3.2 Challenge and Technical Overview

Despite the great characteristics of a mixture model, it is still challenging for us to use it for deriving generalizable insights or individual explanation. This is because a regression mixture model does not always guarantee a success in model approximation, especially when it deals with samples with diverse feature correlations and data sparsity.

To tackle this challenge, an instinctive reaction is to introduce an elastic net to a Bayesian regression mixture model. Past research [9, 18, 38] has demonstrated that an elastic net encourages the grouping effects among variables so that highly correlated variables tend to be in or out of a mixture model together. Therefore, it can potentially augment the aforementioned method with the ability of dealing with the situation where the features of a high dimensional sample are highly correlated. However, a key limitation of this approach could manifest, especially when it deals with samples with diverse feature correlation and data sparsity.

In the following, we address this issue by establishing a dirichlet process mixture model with multiple elastic nets (`DMM-MEN`). Different from previous research [35], our approach allows the regularization terms to has the flexibility to reduce a lasso or ridge under some sample categories, while maintaining the properties of the elastic net under other categories. With the multiple elastic nets, the model is able to capture the different levels of feature correlation and sparsity in the data. the In the following, we provide more details of this hierarchical Bayesian non-parametric model.

3.3 Technical Details

Dirichlet Process Regression Mixture Model. As is specified in Equation (1), the amount of Gaussian distributions is infinite, which indicates that there are infinite number of parameters that need to be estimated. In practice, however, the amount of available data samples is limited and therefore it is necessary to restrict the number of distributions. To do this, truncated Dirichlet process prior [11] can be applied, and Equation (1) can be written as

$$y_i|\mathbf{x}_i, \boldsymbol{\Theta} \sim \sum_{j=1}^{J} \pi_j N(y_i \mid \mathbf{x}_i\boldsymbol{\beta}_j, \sigma_j^2). \tag{2}$$

[1]For multi-class classification tasks, this work approximates each class separately, and thus $\mathbf{X}$ denotes the samples in the same class and $g(\mathbf{X})$ represents the corresponding predictions. Given that $\mathbf{y}$ is a probability vector, we conduct logit transformation before fitting a regression mixture model.

Where J is the hyper-parameter that specify the upper bound of the number of mixture components. To estimate the parameters $\mathbf{\Theta}$, a Bayesian non-parametric approach first models $\pi_{1:J}$ through a "stick-breaking" prior process. With such modeling, parameters $\pi_{1:J}$ can then be computed by

$$\pi_j = u_j \prod_{l=1}^{j-1}(1-u_l) \quad \text{for } j = 2, ..., J-1, \tag{3}$$

with $\pi_1 = u_1$ and $\pi_J = 1 - \sum_{l=1}^{J-1} \pi_l$. Here, u_l follows a beta prior distribution, Beta$(1, \alpha)$, parameterized by α, where α can be drawn from Gamma(e, f) with hyperparameters e and f. To make the computation efficient, σ_j^2 is set to follow an inverse Gamma prior, *i.e.*, $\sigma_j^2 \sim$ Inv-Gamma(a, b) with hyperparameters a and b. Given $\sigma_{1:J}^2$, for conventional Bayesian regression mixture model, $\boldsymbol{\beta}_{1:J}$ can be drawn from Gaussian distribution $N(\mathbf{m}_\beta, \sigma_j^2 \mathbf{V}_\beta)$ with hyperparameters $\mathbf{m}_\beta$ and $\mathbf{V}_\beta$.

As is described above, using a mixture model to approximate a learning model, for any data sample we can identify a regression model to best approximate the prediction of that sample. This is due to the fact that a mixture model can be interpreted as arising from a clustering procedure which depends on the underlying latent component indicators $z_{1:n}$. For each observation $(\mathbf{x}_i, y_i)$, $z_i = j$ indicates that the observation was generated from the j-th Gaussian distribution, *i.e.*, $y_i | z_i = j \sim N(\mathbf{x}_i \boldsymbol{\beta}_j, \sigma_j^2)$ with $P(z_i = j) = \pi_j$.

Dirichlet Process Mixture Model with Multiple Elastic Nets. Recall that a conventional mixture model has difficulty not only in dealing with high dimensional data and highly correlated features but also in handling different types of data heterogeneity. We modify the conventional mixture model by resetting the prior distribution of $\boldsymbol{\beta}_{1:J}$ to realize multiple elastic nets. Specifically, we first define mixture distribution

$$P(\boldsymbol{\beta}_j | \lambda_{1,1:K}, \lambda_{2,1:K}, \sigma_j^2) = \sum_{k=1}^{K} w_k f_k(\boldsymbol{\beta}_j | \lambda_{1,k}, \lambda_{2,k}, \sigma_j^2), \tag{4}$$

where K denotes the total number of component distributions, and $w_{1:K}$ represent component probabilities with $\sum_{k=1}^{K} w_k = 1$. Let $w_k' s$ follow a Dirichlet distribution, *i.e.*, $w_1, w_2, \cdots, w_K \sim$ Dir$(1/K)$. Since we add elastic net regularization to the regression coefficient $\boldsymbol{\beta}_{1:J}$, instead of the aforementioned normal distribution, we adopt the Orthant Gaussian distribution as the prior distribution according to [9]. To be specific, each $\boldsymbol{\beta}_k$ follows a Orthant Gaussian prior, whose density function f_k can be defined as

$$f_k\left(\boldsymbol{\beta}_j | \lambda_{1,k}, \lambda_{2,k}, \sigma_j^2\right) \propto \mathbf{\Phi}\left(\frac{-\lambda_{1,k}}{2\sigma\sqrt{\lambda_{2,k}}}\right)^{-p} \times \sum_{\mathbf{Z} \in \mathcal{Z}} N\left(\boldsymbol{\beta}_j \,\middle|\, -\frac{\lambda_{1,k}}{2\lambda_{2,k}}\mathbf{Z}, \frac{\sigma_j^2}{\lambda_{2,k}}\mathbf{I}_p\right) \mathbf{1}(\boldsymbol{\beta}_j \in \mathcal{O}_{\mathbf{Z}}). \tag{5}$$

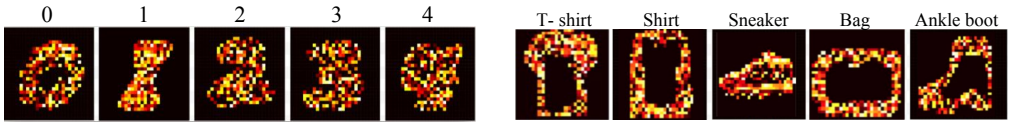
Here, $\lambda_{i,k}$ $(i = 1, 2)$ is a pair of parameters which controls lasso and ridge regression for the k-th component, respectively. We set both to follow Gamma conjugate prior with $\lambda_{1,k} \sim$ Gamma$(R, V/2)$ and $\lambda_{2,k} \sim$ Gamma$(L, V/2)$, where R, L, and V are hyperparameters. $\mathbf{\Phi}(\cdot)$ is the cumulative distribution function of the univariate standard Gaussian distribution, and $\mathcal{Z} = \{-1, +1\}^p$ is a collection of all possible p-vectors with elements ± 1. Let $Z_l = 1$ for $\beta_{jl} \geq 0$ and $Z_l = -1$ for $\beta_{jl} < 0$. Then, $\mathcal{O}_{\mathbf{Z}} \subset \mathbb{R}^p$ can be determined by vector $\mathbf{Z} \in \mathcal{Z}$, indicating the corresponding orthant.

Given the prior distribution of f_k defined in (5), it is difficult to compute the posterior distribution and sample from it. To obtain a simpler form, we use the mixture representation of the prior distribution (5). To be specific, we introduce a latent variable $\boldsymbol{\tau}_{1:p}$ and rewrite the (5) into the following hierarchical form[2]

$$\boldsymbol{\beta}_j \mid \boldsymbol{\tau}_j, \sigma_j^2, \lambda_{2,c_j} \sim N\left(\boldsymbol{\beta}_j \,\middle|\, 0, \frac{\sigma_j^2}{\lambda_{2,c_j}}\mathbf{S}_{\boldsymbol{\tau}_j}\right), \text{and} \tag{6}$$

$$\boldsymbol{\tau}_j \mid \sigma_j^2, \lambda_{1,c_j}, \lambda_{2,c_j} \sim \prod_{l=1}^{p} \text{Inv-Gamma}_{(0,1)}\left(\tau_{jl} \,\middle|\, \frac{1}{2}, \frac{1}{2}\left(\frac{\lambda_{1,c_j}}{2\sigma_j\sqrt{\lambda_{2,c_j}}}\right)^2\right), \tag{7}$$

[2]More details about the derivation of the scale mixture representation and the proof of equivalence can be found in [9, 18].



(a) Generalizable insights extracted from MLP. (b) Generalizable insights extracted from CNNs.

Figure 1: The illustration of Generalizable insights extracted from the MLP trained for recognizing handwritten digits and the CNNs for fitting the Fashion-MNIST dataset. Each pattern contains 150 pixels, the importance of which is illustrated by the heat map. Due to the space limit, the results of other categories are shown in supplementary material.

where $\boldsymbol{\tau}_j \in \mathbb{R}^p$ denotes latent variables and $\mathbf{S}_{\boldsymbol{\tau}_j} \in \mathbb{R}^{p\times p}$, with $\mathbf{S}_{\boldsymbol{\tau}_j} = \text{diag}(1-\tau_{jl})$ for $l = 1, \cdots, p$. Similar to component indicator z_i introduced in the previous section, here, we introduce a set of latent regularization indicators $c_{1:J}$. For each parameter β_j, $c_j = k$ indicates that parameter follows distribution $f_k(\cdot)$ with $P(c_j = k) = w_k$.

Posterior Computation and Post-MCMC Analysis. We develop a customized MCMC method involving a combination of Gibbs sampling and Metropolis-Hastings algorithm for parameter inference [28]. Basically, it involves augmentation of the model parameter space by the aforementioned mixture component indicators $z_{1:n}$ and $c_{1:J}$. These indicators enable simulation of relevant conditional distributions for model parameters. As the MCMC proceeds, they can be estimated from relevant conditional posteriors and thus we can jointly obtain posterior simulations for model parameters and mixture component indicators. We provide the details of posterior distribution and the implementation of updating the parameters in the supplementary material. Considering that fitting a mixture model with MCMC suffers from the well-known label switching problem, we use an iterative relabeling algorithm introduced in [3].

4 Evaluation

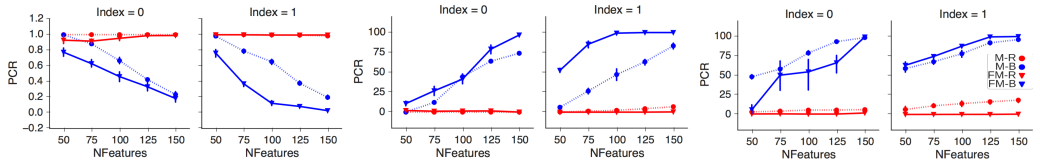
Recall that the motivation of our proposed method is to increase the transparency for complex ML models so that users could leverage our approach to not only understand an individual decision (explainability) but also to obtain insights into the strength and vulnerabilities of the target model (scrutability). The experimental evaluation of the proposed method thus focuses on the aforementioned two aspects – scrutability and explainability.

4.1 Scrutability

Methodology. As a first step, we utilize Keras [2] to train an MLP on MNIST dataset [16] and CNNs to classify clothing images in Fashion-MNIST dataset [34] respectively. These machine learning methods represent the techniques most commonly used for the corresponding classification tasks. We trained these model to achieve more than decent classification performance. We then treat these two models as our target models and apply our proposed approach to establish scrutability.

We define the scrutability of an explanation method as the ability to distill generalizable insights from the model under examination. In this work, generalizable insights refer to feature importance inferences that could be generalized across all cases. Admittedly, the fidelity of our proposed solution to the target model is an important prerequisite to any generalizable insights our solution extracts. In this section, we carry out experiments to empirically evaluate the fidelity while also demonstrating scrutability of our solution. We apply the following procedures to obtain experimentation data.

1. Construct bootstrapped samples from the training data and nullify the top important pixels identified by our approach among positive cases while replacing the same pixels in negative cases with the mean value of those features among positive samples.
2. Apply random pixel nullification/replacement to the same bootstrapped samples used in previous step from the training data.
3. Construct test cases that register positive properties for the top important pixels while randomly assign value for the remaining pixels.



(a) Bootstrapped positive samples. (b) Bootstrapped negative samples. (c) New testing cases.

Figure 2: Results of fidelity validation. Note that `PCR` in y-axis denotes positive classification rate and `NFeature` in x-axis refers to number of features. In the legend, `B` indicates selecting features through our Bayesian approach and `R` represents selecting features through random pick. `M` and `FM` denote datasets MNIST and Fashion-MNIST respectively. Due to the space limit, the results of other categories are shown in supplementary material.

4. Construct randomly created test cases (*i.e.*, assigning random value to all pixels) as baseline samples for the new test cases.

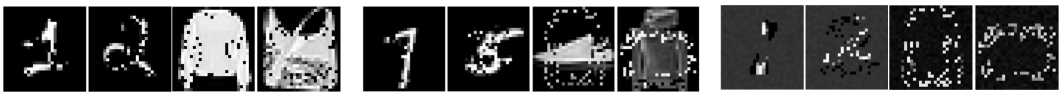
We then compare the target model classification performance among synthetic samples crafted via procedures mentioned above. The intuition behind this exercise is that if the fidelity/scrutability of our proposed solution holds, we should be able to see significant impact on the classification accuracy. Moreover, the magnitude of the impact should significantly outweigh that observed from randomly manipulating features. In the following, we describe our experiment tactics and findings in greater details.

Experimental Results. Figure 1 illustrates the generalizable insights (*i.e.*, important pixels in MNIST and Fashion-MNIST datasets) that our proposed solution distilled from the target MLP and CNNs models, respectively. To validate the faithfulness of these insights and establish fidelity of our proposed solution, we conduct the following experiment.

First, bootstrapped samples, each contains a random draw of 30% of the original cases, are constructed from the MNIST and Fashion-MNIST datasets. For cases that are originally identified as positive for corresponding classes by the target models (*i.e.*, MLP and CNNs), we nullify top 50/75/100/125/150 important features identified by our proposed solution respectively, while forcing the value of corresponding features in the negative samples equal to the mean value of those among the positive samples. These manipulated cases are then supplied to the the target model and we measure the proportion of cases that those models would classify as positive under each condition. In addition, we apply the same perturbations on randomly selected 50/75/100/125/150 features in the same bootstrapped sample and measure the target model's positive classification rate after the manipulation as a baseline for comparison. We repeat such a process for 50 times for both datasets to account for the statistical uncertainty in the measured classification rate.

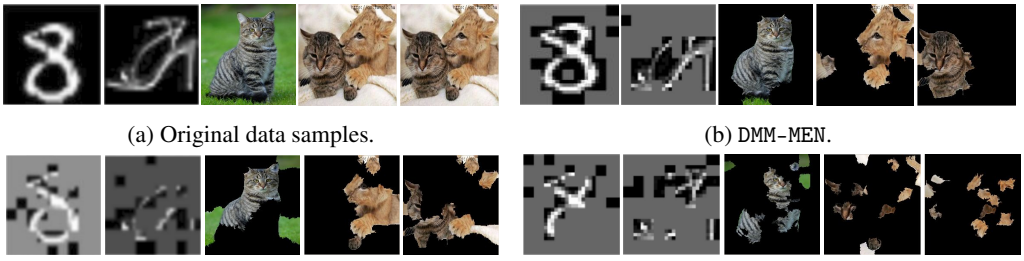
Figure 3a, Figure 3b and supplementary material showcase some of the aforementioned bootstrapped samples. Figure 2a and Figure 2b summarize the experimental results we obtain from the procedures mentioned above. As is illustrated in both figures, the classification rates of the target models on these perturbed samples are impacted dramatically once we start manipulating top 50/75 important features (*i.e.*, around 9% of the pixels in each image) identified by our proposed solution in these images. However, we do not observe any significant impact to the model's classification performance if we randomly perturb the same number of pixels. Non-overlapping 95% confidence intervals of the post-manipulation classification performance also reveal that the impact of these top features is significantly greater than the features selected at random. Moreover, the fact that we start observing dramatic impact in the target models' classification performance after we manipulate less than 9% of the total features justifies the faithfulness of our proposed approach to the ML models under examination.

To further validate the fidelity of the insights illustrated in Figure 1, we construct new testing cases based on top 50/75/100/125/150 pixels deemed important by our proposed solution respectively and measure the proportion of these testing samples that are classified as positive cases by the target models. We also create testing cases by randomly filling 50/75/100/125/150 pixels within the images and measure the positive classification rate as a baseline. The intuition behind this exercise is that, similar to the experiments described earlier, we would like to see significantly higher positive classification rates leveraging the insights from our proposed solution than creating cases around randomly selected pixels. In Figure 3c and supplementary material, we showcase some insights



(a) Bootstrapped positive samples. (b) Bootstrapped negative samples. (c) New cases samples.

Figure 3: Samples manipulated or crafted for scrutability evaluation.



(a) Original data samples. (b) DMM-MEN.

(c) LIME. (d) SHAP.

Figure 4: The examples explaining individual predictions obtained from MLP and CNNs. It should be noted that, since the images in MNIST and Fashion-MNIST has black background, to better illustrate the difference, we change segments of these images to grey if they are not selected.

driven testing cases. As is shown in Figure 2c, insights driven testing cases have much higher success rates than the cases created around random pixels. In fact, we observe that even if we randomly fill 150 pixels (which is close to 20% of the pixels in an image), the positive classification rate remains extremely low across classes. On the contrary, we notice that with the cases created based on the top 50 important pixels (*i.e.*, 9% of all pixels in an image) deemed by our solution, we could already achieve around 50% success rate. For some specific outcome categories, we could even achieve a much higher success rate.

It is worth noting that aforementioned experiments also unveil the vulnerabilities and sensitivities of the target MLP and CNNs models. It does not seem to matter if a handwritten digit or a fashion product is visually recognizable in an image, the model will classify it to the corresponding category with a high confidence as long as the important features indicated in the heat map are filled with greater values (see Figure 3b). In other words, both the MLP and CNNs models evaluated in this study are very sensitive to these pixels but could also be vulnerable to pathological image samples crafted based on such insights. Figure 3a and Figure 3c are two additional examples. A sample (Figure 3a) might carry the right semantics, the learning model still might be blind to that sample if the pixels corresponding to important features are filled with smaller values. On the other hand, a very noisy sample (Figure 3c) could still be correctly classified as long as the pixels corresponding to important features are assigned with decent values.

4.2 Explainability

Our proposed solution does not only extract generalizable insights from the target models but also demonstrate superior performance in explaining individual decisions. To illustrate its superiority, we compare our approach with a couple of state-of-the-art explainable approaches, namely LIME and SHAP. In particular, we evaluate the explainability of these approaches by comparing the explanation feature maps and more importantly quantitatively measuring their relative superiority in identifying influential features in individual decisions.

As is introduced in the aforementioned section, we also evaluate the explainability of our proposed solution on the VGG16 model [27] trained from ImageNet dataset [5]. Due to the ultra high dimensionality concern, which we will discuss in the following section, we adopt the methodology in [23] to generate data to explain individual decisions. More specifically, we create a new dataset by randomly sampling around the data sample that needed to be explained, reducing the dimensionality of the newly crafted dataset by certain dimension reduction method [23] and fitting the approximation model.

Table 1: Quantitative evaluation results of explainability

	DMM-MEN		LIME		SHAP	
	Probability (Confidence Interval)	Accuracy	Probability (Confidence Interval)	Accuracy	Probability (Confidence Interval)	Accuracy
MNIST	99.89% (99.74%, 100%)	100%	99.84% (99.69%, 100%)	99.95%	94.01% (93.99%, 94.03%)	94.10%
Fashion-MNIST	97.59% (97.32%, 97.89%)	100%	93.49% (92.92%, 94.07%)	98.32%	86.03% (85.23%, 86.65%)	90.10%
ImageNet	69.36% (47.88%, 90.18%)	85.6%	47.46% (31.34%, 68.58%)	66.05%	7.85% (5.88%, 28.82%)	14.20%

Figure 4a and supplementary material illustrate ten handwritten digits and ten fashion products randomly selected from each of the classes in MNIST and Fashion-MNIST datasets, respectively. We apply our solution as well as LIME and SHAP to each of the images shown in the figure and then select and highlight the top 20 segments that each approach deems important to the decision made by deep neural network classifiers. The results are presented in Figure 4b, Figure 4c, Figure 4d and supplementary material for our approach, LIME and SHAP, respectively. As we can observe in these figures, our approach nearly perfectly highlights the contour of each digit and fashion product, whereas LIME and SHAP identify only the partial contour of each digit and product and select more background parts than our approach.

Figure 4a also has two images we randomly selected from ImageNet dataset. The left image has only one object and the other image has two. Figure 4b to Figure 4d demonstrate the top 10 segments pinpointed by three explanation techniques. The results shown in these figures are consistent with those of MNIST and Fashion-MNIST. More specifically, the proposed approach can precisely highlight the object in the images, while the other approaches only partly identify the object and even select some background noise as important features. In order to evaluate the fidelity of these explanation results, we input these feature images back to VGG16 and record the prediction probabilities of the true labels (tiger cat, lion and tiger cat). Figure 4b achieved the highest probabilities on each feature map, which from the left to right are 93.20%, 78.51% and 92.70%. Note that in the fourth image of Figure 4b, while identifying a lion in the image, our approach highlights the moustache of the cat, which seems like a wrong selection. However, if we exclude this part from the image, the probability of the object belonging to lion drops from 78.51% to 20.31%. This result showcases a false positive of VGG16 and indicates that we can still find the weakness of the target model even from the individual explanations.

To further quantify the relative performance in explainability, we also conduct the following experiment. First we randomly select 10000 data samples from aforementioned datasets. Then, we apply our approach as well as two state-of-the-art solutions (*i.e.*, LIME and SHAP) to extract top 20 important segments (top 10 segments for ImageNet dataset). We then manipulate these samples based on the segments identified via three approaches. To be specific, we only keep the top important pixels intact while nullifying the remaining pixels and supply these manipulated samples to the target models and evaluate the classification accuracy. Table 1 shows the accuracy of these feature images being classified to the corresponding truth categories as well as the means and the 95% confidence interval of the prediction probabilities. The results indicate that our approach offers better resolution and more granular explanations to individual predictions. One possible explanation is that both LIME and SHAP assume the local decision boundary of the target model to be linear while the proposed approach conducts the variable selection by applying a non-linear approximation.

It is known that Bayesian non-parametric models are computationally expensive. However, It does not mean that we cannot use the proposed approach in the real-world applications. In fact, we have recorded the latency of the proposed approach on explaining individual samples in three datasets. The running times are for MNIST, Fashion-MNIST and ImageNet are 37.5s, 44s and 139.2s, respectively. As to approximating the global decision boundary, the running times are 105 mins on MNIST and 115 mins on Fashion-MNIST. It is believed that the latency of our approach is still within the range of normal training time for complex ML models.

5 Discussion

Scalability. As is shown in Section 4, our proposed solution does not impose incremental challenge on scalability. We can still further accelerate the algorithm to improve its scalability. More specifically,

current advances in Bayesian computation approaches allow the MCMC methods to be used for big data analysis, such as adopting Bootstrap Metropolis–Hastings Algorithm [19], applying divide and conquer approaches [30] and even taking advantage of GPU programming to speed up the computation [31].

Data Dimensionality. Our evaluation described in Section 4 indicates that the proposed solution (`DMM-MEN`) could extract generalizable insights even from high dimensional data (*e.g.* Fashion MNIST). However, when it comes to ultra high-dimensional data, getting generalizable insights could still be a challenge. One obvious reason is that we do not have sufficient data to infer all the parameters. More importantly, even if we had enough data, it would be very computationally expensive. Arguably, one solution is to reduce the dimensionality of such ultra high dimensional data while preserving the original data distribution. However, take ImageNet dataset as an example. Even the state-of-the-art dimensionality reduction methods (*i.e.,* the one used in [23]) could not satisfactorily preserve the whole data distribution. This indeed speaks to the limitation of our proposed solution in extracting generalizable insights when it comes to specific datasets. Nevertheless, it does not affect our solution's ability in precisely explaining individual predictions even when it comes to ultra high dimensional data. As is shown in Section 4, our solution significantly outperforms the state-of-the-art solutions in explaining individual decisions made on ultra-high dimensional data samples.

Other Applications and Learning Models. While we evaluate and demonstrate the capability of our proposed technique only on the image recognition using deep learning models, the proposed approach is not limited to such a learning task and models. In fact, we also evaluated our technique on other learning tasks with various learning models. We observed the consistent superiority in extracting global insight and explaining individual decisions. Due to the space limit, we specify those experiment results in our supplementary material submitted along with this manuscript.

6 Conclusion and Future Work

This work introduces a new technical approach to derive generalizable insights for complicated ML models. Technically, it treats a target ML model as a black box and approximates its decision boundary through `DMM-MEN`. With this approach, model developers and users can approximate complex ML models with low errors and obtain better explanations of individual decisions. More importantly, they can extract generalizable insights learned by a target model and use it to scrutinize model strengths and weaknesses. While our proposed approach exhibits outstanding performance in explaining individual decisions, and provides a user with an ability to discover model weaknesses, its performance may not be good enough when applied to interpreting temporal learning models (*e.g.*, recurrent neural networks). This is due to the fact that, our approach takes features independently whereas time series analysis deals with features temporally dependent. As part of the future work, we will therefore equip our approach with the ability of dissecting temporal learning models.

Acknowledgments We gratefully acknowledge the funding from NSF grant CNS-1718459 and the support of NVIDIA Corporation with the donation of the GPU. We also would like to thank anonymous reviewers, Kaixuan Zhang, Xinran Li and Chenxin Ma for their helpful comments.

References

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

[2] F. Chollet et al. Keras, 2015.

[3] A. J. Cron and M. West. Efficient classification-based relabeling in mixture models. *The American Statistician*, 2011.

[4] P. Dabkowski and Y. Gal. Real time image saliency for black box classifiers. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the 22nd Conference on Computer Vision and Pattern Recognition. (CVPR)*, 2009.

[6] R. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the 16th International Conference on Computer Vision (ICCV)*, 2017.

[7] N. Frosst and G. Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[8] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann. Devnet: A deep event network for multimedia event detection and evidence recounting. In *Proceedings of the 28th Conference on Computer Vision and Pattern Recognition. (CVPR)*, 2015.

[9] C. Hans. Elastic net regression modeling with the orthant normal prior. *Journal of the American Statistical Association*, 2011.

[10] C. Hennig. Methods for merging gaussian mixture components. *Advances in data analysis and classification*, 2010.

[11] H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 2001.

[12] B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, 2016.

[13] W. Knight. The dark secret at the heart of ai. `https://www.technologyreview.com/s/604087/`, 2017.

[14] W. Knight. The financial world wants to open ai's black boxes. `https://www.technologyreview.com/s/604122/`, 2017.

[15] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[16] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.

[17] J. Li, W. Monroe, and D. Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

[18] Q. Li, N. Lin, et al. The bayesian elastic net. *Bayesian Analysis*, 2010.

[19] F. Liang, J. Kim, and Q. Song. A bootstrap metropolis–hastings algorithm for bayesian analysis of big data. *Technometrics*, 2016.

[20] Z. C. Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[21] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[22] J.-M. Marin, K. Mengersen, and C. P. Robert. Bayesian modelling and inference on mixtures of distributions. *Handbook of statistics*, 2005.

[23] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[24] 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. *arxiv. org/abs/1610.02391 v3*, 2016.

[25] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[26] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[28] A. Smith and G. Roberts. Bayesian computation via the gibbs sampler and related markov chain monte carlo methods. *Journal of the Royal Statistical Society. Series B*, pages 3–23, 1993.

[29] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *Proceedings of the 3rd International Conference on Learning Representations Workshop (ICLR Workshop)*, 2015.

[30] S. Srivastava, V. Cevher, Q. Dinh, and D. Dunson. Wasp: Scalable bayes via barycenters of subset posteriors. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

[31] M. Suchard, Q. Wang, C. Chan, F. J., A. Cron, and M. West. Understanding gpu programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of computational and graphical statistics*, 2010.

[32] M. Sundararajan, A. Taly, and Q. Yan. Gradients of counterfactuals. *arXiv preprint arXiv:1611.02639*, 2016.

[33] M. Wu, M. C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[34] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[35] H. Yang, D. Dunson, and D. Banks. The multiple bayesian elastic net. *submitted for publication*, 2011.

[36] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proceedings of the 13rd European Conference on Computer Vision (ECCV)*, 2014.

[37] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[38] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2005.

Supplementary Material: Explaining Deep Learning Models – A Bayesian Non-parametric Approach

1 Implementation

1.1 Posterior Distribution

We present the full posterior of the proposed approach as follows. The following parameters are updated via Gibbs Sampling. First,

$$P(z_i = j \mid \mathbf{x}_i, y_i, \boldsymbol{\beta}_j, \sigma_j^2) \propto P(y_i \mid \mathbf{x}_i, \boldsymbol{\beta}_j, \boldsymbol{\tau}_j, \sigma_j^2) P(z_i = j)\,, \tag{1}$$

and

$$u_j \Big| \mathbf{X}, \mathbf{y} \sim \text{Beta}(a_j + 1, \alpha + \sum_{l=j+1}^{J} a_l)\,, \tag{2}$$

where $a_j = \sum_{i=1}^{N} \mathbf{1}(z_i = j)$, for $j = 1, \cdots, J$ with $\mathbf{1}$ the indicator function. Then we let

$$\alpha \Big| u_{1:J-1} \sim \text{Gamma}(J + e + 1, f - \sum_{j=1}^{J-1} \log(1 - u_j))\,, \tag{3}$$

$$P(c_j = k \mid \boldsymbol{\beta}_j, \sigma_j^2, \lambda_{2,k}) \propto P(\boldsymbol{\beta}_j \mid c_j = k, \sigma_j^2, \lambda_{2,k}) P(c_j = k)\,, \tag{4}$$

$$w_1, w_2, \cdots, w_K \Big| \frac{1}{K} \sim \text{Dir}\left(\frac{1}{K} + b_1 - 1, \frac{1}{K} + b_2 - 1, \cdots, \frac{1}{K} + b_K - 1\right)\,, \tag{5}$$

where $b_k = \sum_{j=1}^{J} \mathbf{1}(c_j = k)$, for $k = 1, \cdots, K$, and

$$\boldsymbol{\beta}_j \mid \boldsymbol{\tau}_j, \lambda_{2,k}, \mathbf{X}, \mathbf{y} \sim N(\mathbf{R}_{\boldsymbol{\tau}_j} \mathbf{X}_{\mathcal{O}_j}^{\mathrm{T}} \mathbf{y}_{\mathcal{O}_j}, \sigma_j^2 \mathbf{R}_{\boldsymbol{\tau}_j})\,, \tag{6}$$

where $\mathbf{R}_{\boldsymbol{\tau}_j} = (\mathbf{X}_{\mathcal{O}_j}^{\mathrm{T}} \mathbf{X}_{\mathcal{O}_j} + \lambda_{2,c_j} \mathbf{S}_{\boldsymbol{\tau}_j}^{-1})^{-1}$. We define $\mathcal{O}_j = \{i | z_i = j\}$. Then, $\mathbf{X}_{\mathcal{O}_j}$ represents the matrix composed by the vectors $\mathbf{x}_i$ whose subscript belongs to the set $\mathcal{O}_j$ and $\mathbf{y}_{\mathcal{O}_j}$ represents the vector, where each element is y_i whose subscript also belongs to the set $\mathcal{O}_j$. By the change of variable, let $s_{jl} = \tau_{jl}/(1 - \tau_{jl})$ to ensure that τ_{jl} falls into the interval $(0, 1)$, then we have

$$s_{jl} \mid \boldsymbol{\beta}_j, \sigma_j^2, \lambda_{1,c_j}, \lambda_{2,c_j} \sim \text{Inv-Gamma}\left(\frac{\lambda_{1,c_j}}{2\lambda_{2,c_j} |\boldsymbol{\beta}_j|}, \frac{\lambda_{1,c_j}^2}{4\lambda_{2,c_j}} \sigma_j^2\right)\,. \tag{7}$$

The other three sets of variables $\sigma_{1:J}^2$, $\lambda_{1,1:K}$, $\lambda_{2,1:K}$ are updated by Metropolis-Hasting Sampling. The posterior and proposal distribution of variables in each set are as follows. First,

$$P(\sigma_j^2 \mid \boldsymbol{\beta}_j, \boldsymbol{\tau}_j, \mathbf{X}, \mathbf{y}) \propto \prod_{i=1}^{a_j} P(y_i \mid \mathbf{x}_i \boldsymbol{\beta}_j, \sigma_j^2) P(\boldsymbol{\beta}_j \mid \sigma_j^2, \boldsymbol{\tau}_j, \lambda_{2,c_j}) P(\boldsymbol{\tau}_j \mid \sigma_j^2, \lambda_{1,c_j}, \lambda_{2,c_j}) P(\sigma_j^2)\,. \tag{8}$$

Then the proposed distribution follows

$$\prod_{i=1}^{a_j} P(y_i \mid \mathbf{x}_i \boldsymbol{\beta}_j, \sigma_j^2) P(\boldsymbol{\beta}_j \mid \sigma_j^2, \boldsymbol{\tau}_j, \lambda_{2,c_j}) P(\sigma_j^2) \propto \text{Inv-Gamma}(a_N, b_N)\,, \tag{9}$$

where $a_N = a + (a_j + 1)/2$ and $b_N = b + (\sum_{i=1}^{a_j} (g_i - \mathbf{x}_i \boldsymbol{\beta}_j)^2 + \boldsymbol{\beta}_j^{\mathrm{T}} \mathbf{S}_{\boldsymbol{\tau}_j}^{-1} \boldsymbol{\beta}_j \lambda_{2,c_j})/2$.

For $\lambda_{1,k}$ and $\lambda_{2,k}$,

$$P(\lambda_{1,k}, \lambda_{2,k} \mid \boldsymbol{\beta}_j, \boldsymbol{\tau}_j) \propto \prod_{j=1}^{b_k} P(\boldsymbol{\beta}_j \mid \boldsymbol{\tau}_j, \lambda_{2,k}, \sigma_j^2) P(\boldsymbol{\tau}_j \mid \lambda_{1,k}, \lambda_{2,k}) P(\lambda_{1,k}, \lambda_{2,k})\,. \tag{10}$$

We use a truncated normal distribution to represent the proposed distribution, that is,

$$\lambda_{1,k} \sim \text{TN}(\lambda_{1,k}, v_{1,N}), \quad \lambda_{2,k} \sim \text{TN}(\lambda_{2,k}, v_{2,N})\,, \tag{11}$$

where TN stands for truncated normal distribution, and $v_{1,N}$ and $v_{2,N}$ are the hyperparameters.

1.2 Training Algorithm

The algorithm of estimate the parameters of the proposed model is shown in Table 1. We implement the algorithm using `R` and the code is available at https://github.com/Henrygwb/dmm-men.

Algorithm: Customized MCMC for Parameter Inference

Input: Data Matrix $\mathbf{X}$, corresponding prediction $\mathbf{y}$, iteration number TT, mixture component number J, elastic nets number K, hyper-parameters e, f, a, b, L, R, V.

Initialization: Sampling $\alpha^{(1)}$, $\mu_{1:J-1}^{(1)}$, $\pi_{1:J-1}^{(1)}$, $z_{1:n}^{(1)}$, $\lambda_{1,1:K}^{(1)}$, $\lambda_{2,1:K}^{(1)}$, $w_{1:K}^{(1)}$, $c_{1:J}^{(1)}$, ${\sigma_{1:J}^2}^{(1)}$, $\boldsymbol{\beta}_{1:J}^{(1)}$, $\boldsymbol{\tau}_{1:J}^{(1)}$ from the prior distributions introduced in the paper.

for $tt = 2, ..., TT$ **do**

Updating the following parameters with Gibbs sampling:

$z_{1:n}^{(tt)}$,
$\mu_{1:J-1}^{(tt)}$,
$\pi_{1:J-1}^{(tt)}$,
$\alpha^{(tt)}$,
$c_{1:J}^{(tt)}$,
$w_{1:K}^{(tt)}$,
$\boldsymbol{\beta}_{1:J}^{(tt)}$,
$\boldsymbol{s}_{1:J}^{(tt)}$ and $\boldsymbol{\tau}_{1:J}^{(tt)}$.

Updating the following parameters by Metropolis-Hasting sampling:

${\sigma_{1:J}^2}^{(tt)}$,
$\lambda_{1,1:K}^{(tt)}$ and $\lambda_{2,1:K}^{(tt)}$.

end for

Output: $\alpha^{(TT)}$, $\mu_{1:J-1}^{(TT)}$, $\pi_{1:J-1}^{(TT)}$, $z_{1:n}^{(TT)}$, $\lambda_{1,1:K}^{(TT)}$, $\lambda_{2,1:K}^{(TT)}$, $w_{1:K}^{(TT)}$, $c_{1:J}^{(TT)}$, ${\sigma_{1:J}^2}^{(TT)}$, $\boldsymbol{\beta}_{1:J}^{(TT)}$, $\boldsymbol{\tau}_{1:J}^{(TT)}$.

Table 1: The proposed customized MCMC algorithm to train the `DMM-MEN` model. Note that the details of posterior distributions for Gibbs sampling and the proposal distributions for Metropolis-Hasting sampling can be found in the Section 1.1.

	Neural Network Structure	Activation	Optimizer	Learning Rate	Regularization	Batch	Epoch
DNN	784-512-512-10	Relu	RMSprop	0.001	Dropout (0.2)	128	50
CNN	Shown in Table 3	Relu	RMSprop	0.001	×	500	30

Table 2: Hyperparameters of DNN trained from MNIST and CNN trained from Fashion-MNIST.

Layer Type	CNN Architecture
Convolotional	32 filters (5×5)
Max Pooling	2×2
Convolotional	64 filters (5×5)
Max Pooling	2×2
Softmax	10

Table 3: Architecture of CNN model. Note that we build the model according to one of baselines shown in [3].

2 Hyperparameters

2.1 Hyperparameters of target models

The hyperparameters of the MLP trained on MNIST dataset and the CNNs trained to classify fashion products in Fashion-MNIST dataset are shown in Table 2. Note that we trained the target deep learning models to achieve the state-of-the-art classification performance on the original datasets, the test accuracy of which are 98.32% on MNIST and 91.24% on Fashion-MNIST.

2.2 Hyperparameters of proposed technology

The hyperparameters of the proposed `DMM-MEN` on each target machine learning model are shown in Table 4.

Datasets	Target models	J	e	f	K	R	L	V	a	b	v_{1_N}	v_{2_N}
'comp.sys'	Random Forest	5	1	1	3	4	1	1	0.5	0.5	2	2
	SVM	5	5	1	3	5	1	1	1	1	2	2
MNIST	MLP	6	5	1	3	2.5	1	1	0.5	0.5	2	2
Fashion-MNIST	CNNs	10	5	1	3	2.5	1	1	0.5	0.5	2	2
ImageNet	CNNs	10	5	1	3	2.5	1	1	0.5	0.5	2	2

Table 4: Hyperparameters of proposed methods for all the target models

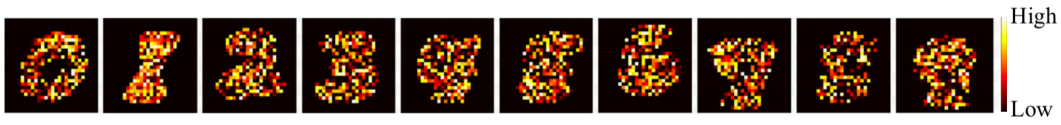
In Table 4, J is the upper bound of the mixture components, e and f are the hyperparameters of Gamma(e, f) and a, b are the hyperparameters of Inv-Gamma(a, b). K is the total number of elastic-net. R, L, V are the hyperparameters of Gamma(R, $V/2$) and Gamma(L, $V/2$). v_{1_N} and v_{2_N} are the hyperparameters of the truncated normal distribution in (11).

2.3 Experimental Results on Image Recognition.

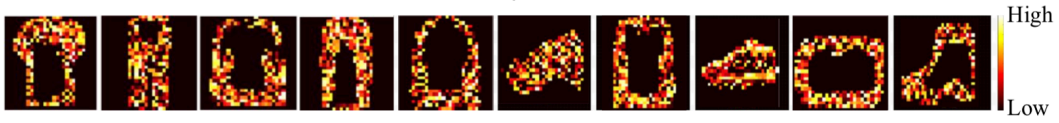
2.4 Scrutability

Figure 1 demonstrates the generalizable insights for all of the categories in MNIST and Fashion-MNIST. Similar to **Section 4**, we highlights the most important 150 pixels in each category. Figure 7 shows the fidelity test results of all of the categories.

Figure 8 - Figure 10 showcase more examples of fidelity testing samples generated from MNIST dataset. Samples generated from Fashion-MNIST are shown in Figure 11 - Figure 13. Actually, we can automatically generate a large bunch of fidelity testing samples according to the method introduced in the **Section 4**. These samples can not only be used to evaluate the fidelity of the proposed approach, but also serve as adversarial samples (*i.e.*, Bootstrapped positive samples shown in 8 and 11) and pathology samples (*i.e.*, Bootstrapped negative samples shown in 9 and 12 and new



(a) Generalizable insights extracted from MLP.



(b) Generalizable insights extracted from CNNs.

Figure 1: The illustration of Generalizable insights extracted from the MLP trained for recognizing handwritten digits and the CNNs for fitting the Fashion-MNIST dataset.

(a) Handwritten digits randomly selected from MNIST and Fashion-MNIST datasets.

(b) Most influential pixels highlighted by DMM-MEN.

(c) Most influential pixels highlighted by LIME.

(d) Most influential pixels highlighted by SHAP.

Figure 2: The examples explaining individual predictions obtained from MLP and CNN. It should note that, to better illustrate the difference, we change pixels in gray if they are not selected.

testing samples shown in 10 and 13). To be specific, adversarial samples refer to samples that carry the right semantics but be classified to the wrong class by the target learning models. Pathology samples are samples that being correctly classified by the target learning models but do not contains the correct objects. Note that both of these samples explore the weakness of the target models; meanwhile the adversarial samples can be adopted to retrain the target model and improve its robustness.

2.5 Explainability

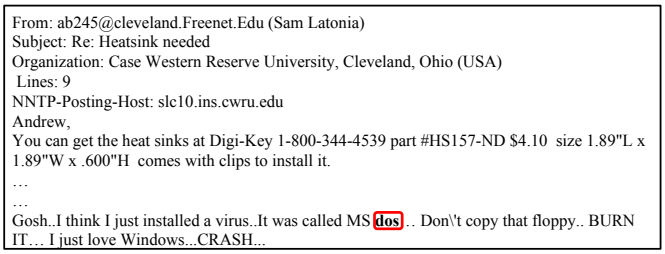
Figure 2 demonstrates more explanation results of MNIST and Fashion-MNIST datasets.

Note that in our evaluation, the LIME established better explanability than SHAP, which does not align with the theoretical results in [2]. The reason is that SHAP guarantees the best results by exploring nearly all of the possible feature combinations in the feature space. However, in practice, this is extremely hard to accomplish. In their implementation, SHAP conduct a limit number of combinations searching. It is highly likely that within this number of searching, SHAP is still not able to identify optimal combination of important features.

It is known that Bayesian non-parametric models are computationally expensive. However, It does not mean that we cannot use the proposed approach in the real-world applications. In fact, we have recorded the latency of the proposed approach on explaining individual samples in three datasets. The running times are for MNIST, Fashion-MNIST and ImageNet are 37.5s, 44s and 139.2s, respectively. As to approximating the global decision boundary, the running times are 105 mins on MNIST and 115 mins on Fashion-MNIST. It is believed that the latency of our approach is still within the range of normal training time for complex ML models.

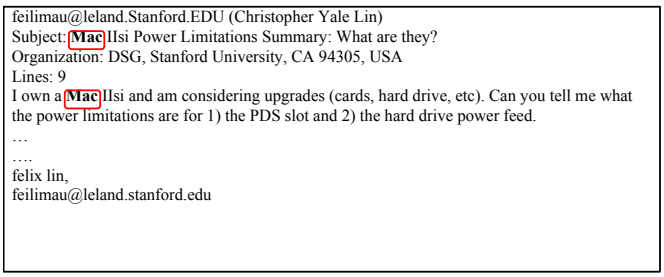
Random Forest		SVM	
'ibm.pc.hardware'	'mac.hardware'	'ibm.pc.hardware'	'mac.hardware'
'dos'	'mac'	'ide'	'mac'
'controller'	'apple'	'gateway'	'Macintosh'
'pc'	'quadra'	'dos'	'quadra'
'windows'	'edu'	'pc'	'apple'

Table 5: The keywords that our approach extracts, indicating the features most influential upon classifications.



	RF	SVM
Original	94.20%	93.47%
mac	10.80%	0.01%
apple	13.00%	1.72%
quadra	16.60%	0.41%
Macintosh	58.00%	0.24%
edu	19.70%	60.47%

Figure 3: An example text snippet categorized into 'ibm.pc.hardware' by random forest (RF) and SVM (SVM). The percentages shown in the table indicate the confidence of being categorized in 'ibm.pc.hardware'.



	RF	SVM
Original	99.40%	99.90%
dos	18.20%	0.87%
controller	20.00%	9.51%
pc	25.40%	1.58%
ide	28.60%	0.02%
gateway	29.92%	0.14%
windows	26.45%	35.36%

Figure 4: An example text snippet categorized into 'mac.hardware' by random forest (RF) and SVM (SVM). The percentages shown in the table indicate the confidence of being categorized in 'mac.hardware'.

3 Experimental Results on Text Mining.

Besides the experiments on MLP and CNNs shown in the **Section 4** of our paper, we also applied our method to machine learning models that are self explainable (*i.e.*, random forest and support vector machines) on text mining. Similar with **Section 4**, we first introduce the dataset used to train the random forest and support vector machines (SVM) models. Then, we demonstrate the scrutability and explainablity of our proposed method. Note that since these target models are self explainable, we donot need to test the fidelity of our proposed method by experiments. Actually, the fidelity of our model can be evaluated by simply comparing the important features extracted by our method with those identified from original models.

The dataset we use is a subset of news20 newsgroups dataset [1]:

'comp.sys' newsgroups dataset [1]: It is a collection of newsgroups posts containing 1,945 samples across 2 topics. The newsgroups posts are split into training and testing datasets based on the dates they have been posted.

3.1 Scrutability

Since SVM and random forest models are explainable by nature, we leverage this case to quantify our solution's faithfulness to a target model. To be specific, we identify top four influential words for each of the classification task using our solution, which are shown in Table 5, and compare them with the top four most weighted features in the original model. While the order of top four influential words vary slightly in our comparison against SVM, the words that our solution identifies match perfectly with those revealed in SVM and random forest model.

Figure 3 shows one classification example, in which both learning models classify the snippet into 'ibm.pc.hardware' with high confidence (94.20% and 93.47% for random forest and SVM, respectively). We replace word '*dos*' – important for both classifiers – with the words shown in Figure 3, and test each of the newly crafted snippets against both classifiers. The value shown in Figure 3 indicates the confidences of categorizing new snippets into 'ibm.pc.hardware'. We notice that by replacing '*dos*' with words that our solution deems important for another class (*i.e.*, 'mac.hardware'), we dramatically reduce the ML model's confidence in classifying the snippet under investigation as 'ibm.pc.hardware'. This again verifies the patterns that our approach extracts accurately reflect what are learned by both ML classifiers.

We also take an text snippet belonging to 'mac.hardware' – shown in Figure 4. In this example, both learning models classify the snippet into 'mac.hardware' with high confidence (99.40% and 99.90% for random forest and SVM, respectively). The words shown in Figure 4 are used to replace word '*Mac*'. We also test each of the newly crafted snippets against both classifiers. The value shown in Figure 4 indicates the confidences of categorizing new snippets into 'ibm.pc.hardware'. Similar to the results of 'mac.hardware', by replacing '*Mac*' with words that our solution deems important for 'mac.hardware', the confidences of being classified to 'ibm.pc.hardware' by ML model's are dramatically reduced. This again also verifies the important words that our approach extracts accurately reflect what are learned by both ML classifiers.

3.2 Explainablity

	Random Forest	SVM
DMM-MEN	**mac apple quadra edu**	**mac Macintosh quadra apple**
LIME	**Macintosh edu apple Re**	**apple SIMM Macintosh about**
SHAP	**mac Macintosh edu apple**	**mac apple Macintosh noah**

From: noah@apple.com (Noah Price)
Subject: Re: How long do RAM SIMM's last?
Distribution: usa
Organization: (not the opinions of) Apple Computer, Inc
Lines: 12
In article <1993Apr11.234818.1755@ultb.isc.rit.edu>,
jek5036@ultb.isc.rit.edu (J.E. King) wrote:
Doesn't a 1 MB SIMM have about 1024 * 1024 * 8 moving flip-flops?
…..
…..
…..
…..
noah
~~~~~~
noah@apple.com Macintosh Hardware Design
...!{sun,decwrl}!apple!noah (not the opinions of) Apple Computer, Inc.

Figure 5: The examples explaining individual predictions obtained from random forest and SVM trained for classifying 'mac.hardware' news posts. Note that the text in bold indicate top-4 keywords most influential upon text classification.

Similar to what we observe in image recognition cases, our approach also outperforms LIME and SHAP in the context of text classification. Figure 5 illustrates one such example: the words highlighted are the most influential indicators for determining if the text snippet belongs to the category of 'mac hardware'. By applying both our approach, LIME and SHAP to the random forest and SVM classifiers, we can observe that the keywords highlighted by our approach is intuitively more distinguishable than those identified by LIME and SHAP.

Figure 6 demonstrates another examples. The words highlighted are the most important words for 'ibm.pc.hardware'. Results also indicate that proposed technology provides more distinguishable key words than those identified by LIME and SHAP. For example, 'SCSI' is a kind of computer interface, which also has been used to Macintosh. Therefore, it can not be treated as an indicator of 'ibm.pc.hardware'.
~~~~~~

	Random Forest	SVM
DMM-MEN	**dos** **controller** **pc** **windows**	**ide** **gateway** **dos** **pc**
LIME	**ide** **dos** **controller** **pc**	**ide** **SCSI** **pc** **windows**
SHAP	**dos** **controller** **ide** **pc**	**ide** **dos** **pc** **windows**

From: guyd@austin.ibm.com (Guy Dawson)
Subject: Re: IDE vs SCSI
Originator: guyd@pal500.austin.ibm.com
rganization: IBM Austin
Lines: 35
In article <1qlbrlINN7rk@dns1.NMSU.Edu>, bgrubb@dante.nmsu.edu (GRUBB) writes: In PC Magazine
April 27, 1993:29 "Although SCSI is twice as fasst as ESDI, 20% faster than IDE, and support up to 7 devices its acceptance …
....
....
....
....
I beleive this last bit is just plain wrong! SCSI-1 intergration is sited as another part of the MicroSoft Plug and play program.
Guy Dawson - Hoskyns Group Plc.

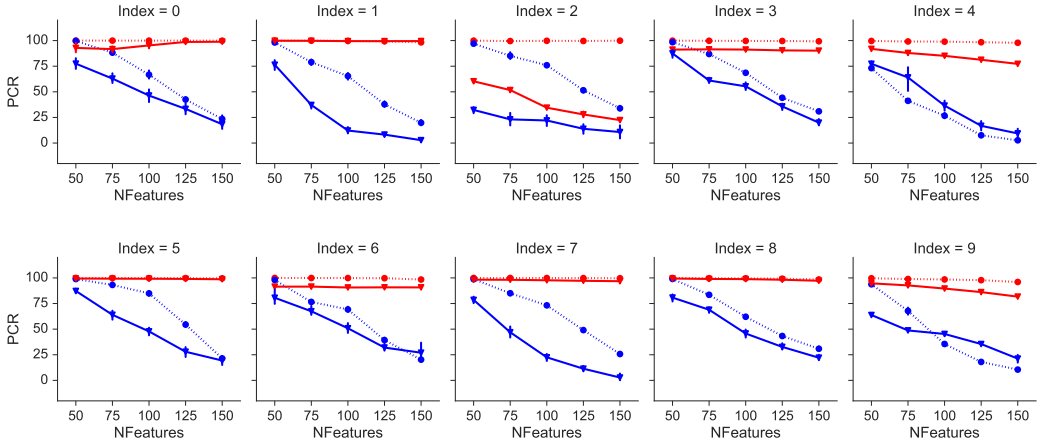
Figure 6: The examples explaining individual predictions obtained from random forest and SVM trained for classifying 'ibm.pc.hardware' news posts.

References

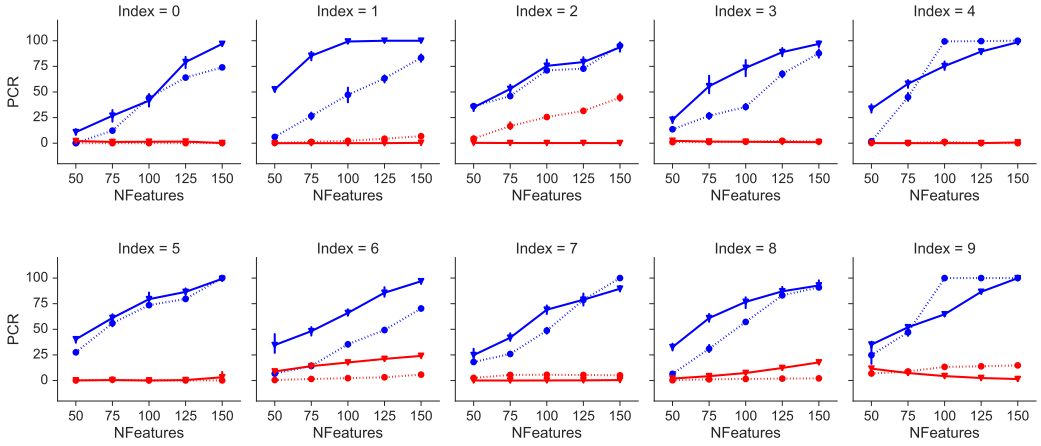
[1] L. K. Newsweeder: Learning to filter netnews. In *Proceedings of the 12nd International Conference on Machine Learning (ICML)*, 1995.

[2] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, 2017.

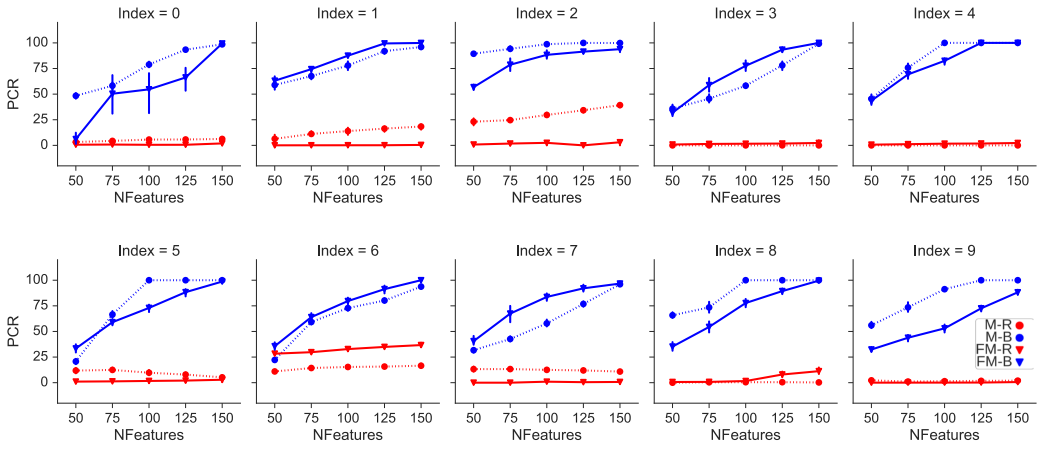
[3] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.



(a) Bootstrapped positive samples.



(b) Bootstrapped negative samples.



(c) New testing cases.

Figure 7: Results of fidelity validation on each category of MNIST and Fashion-MNIST. `PCR` in y-axis denotes positive classification rate and `NFeature` in x-axis refers to number of features. In the legend, `B` indicates selecting features through our Bayesian approach and `R` represents selecting features through random pick. `M` and `FM` denote datasets MNIST and Fashion-MNIST respectively.

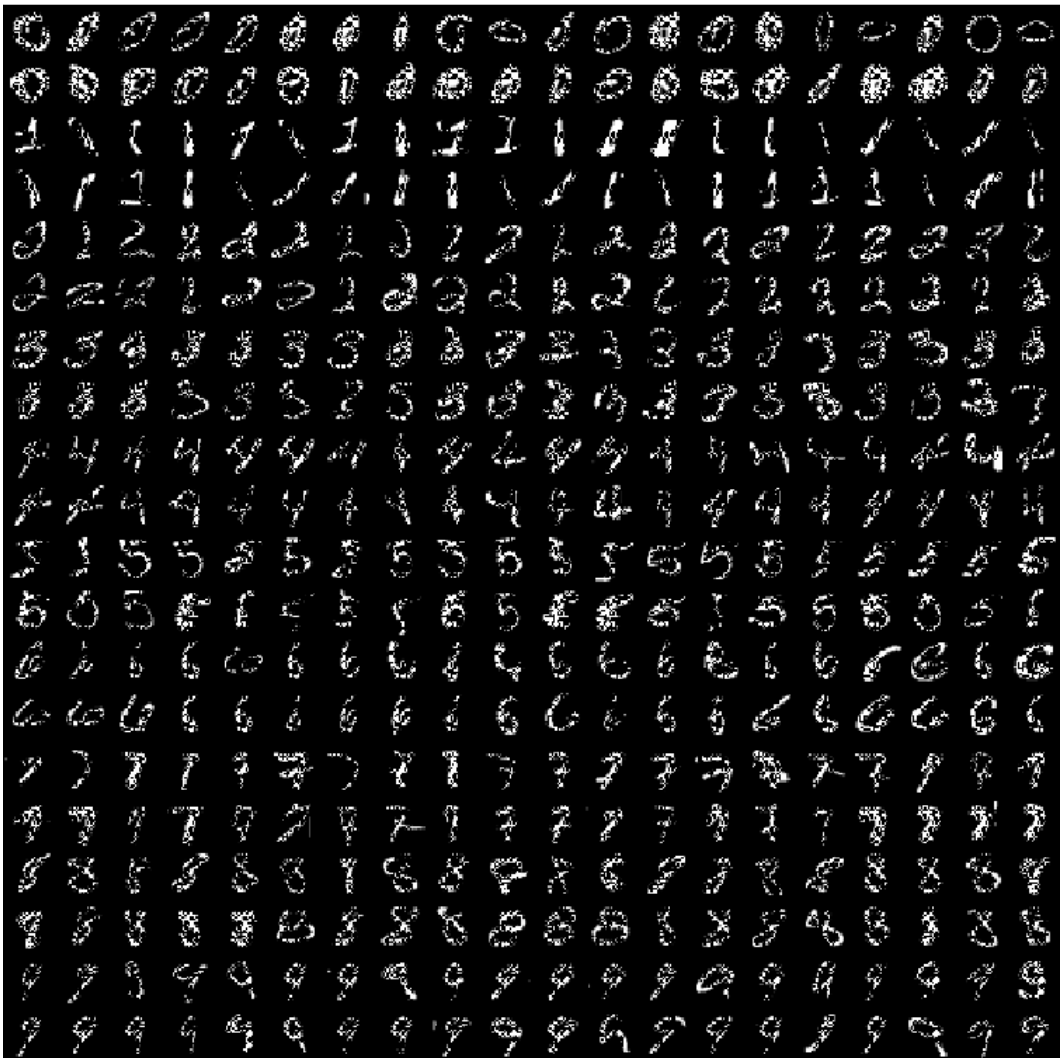


Figure 8: Bootstrapped positive samples of MNIST.

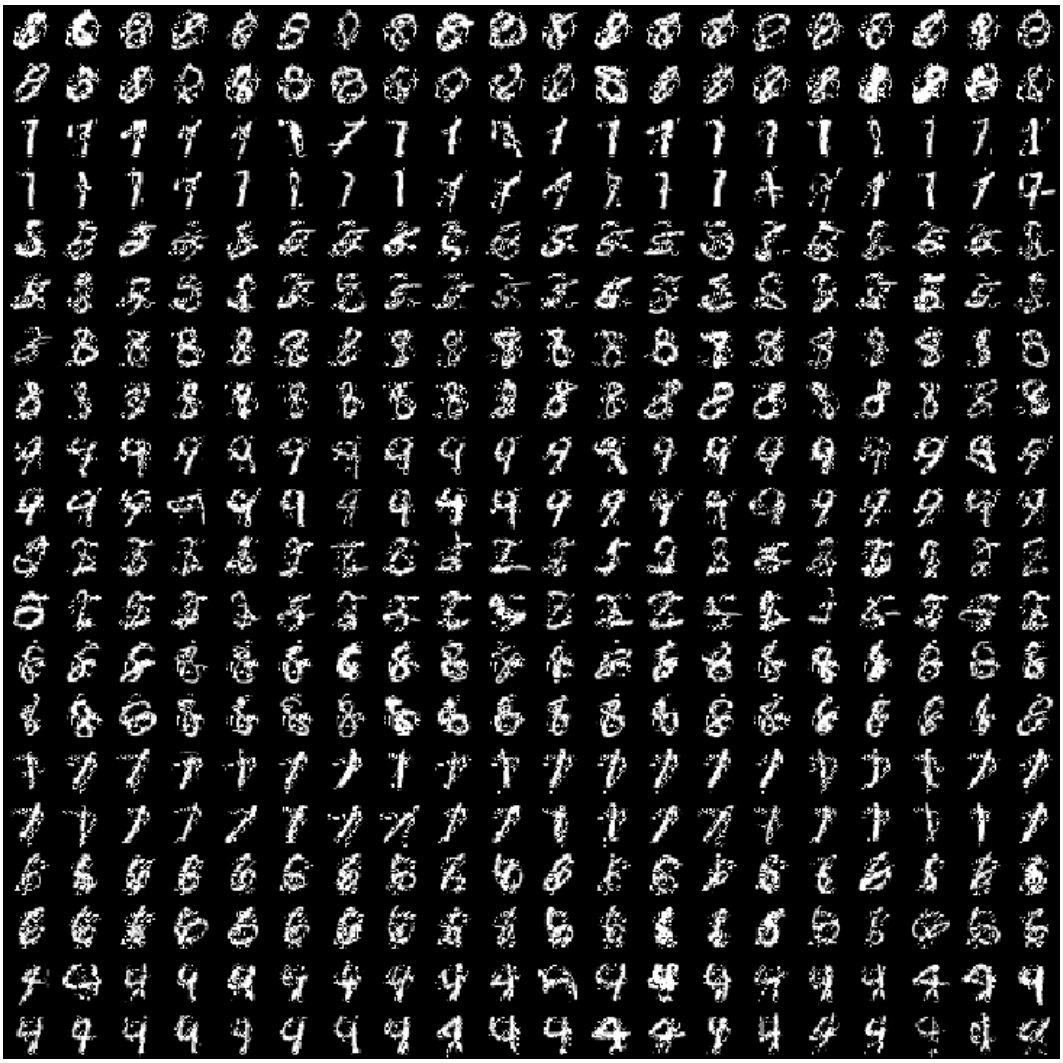


Figure 9: Bootstrapped negative samples of MNIST.

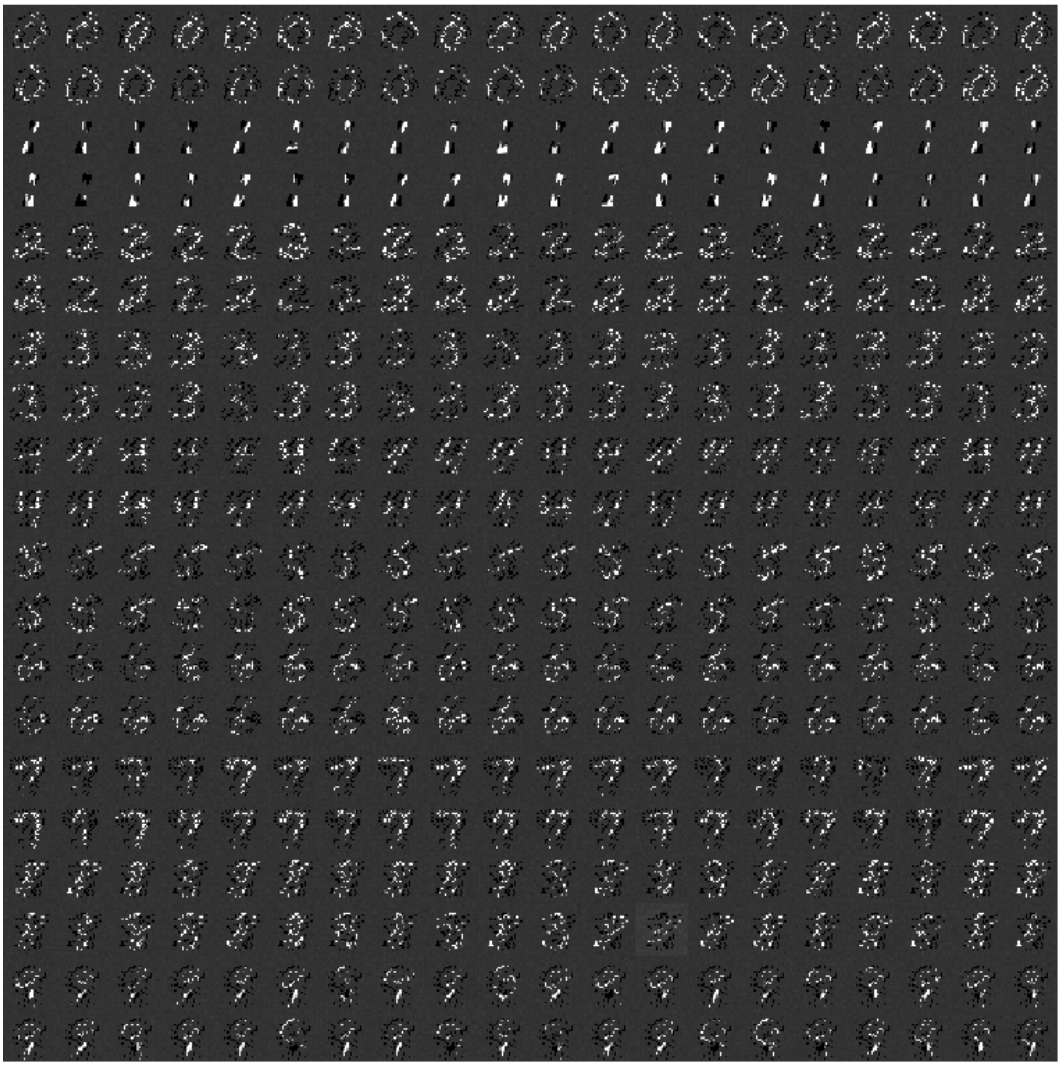


Figure 10: New testing cases of MNIST.

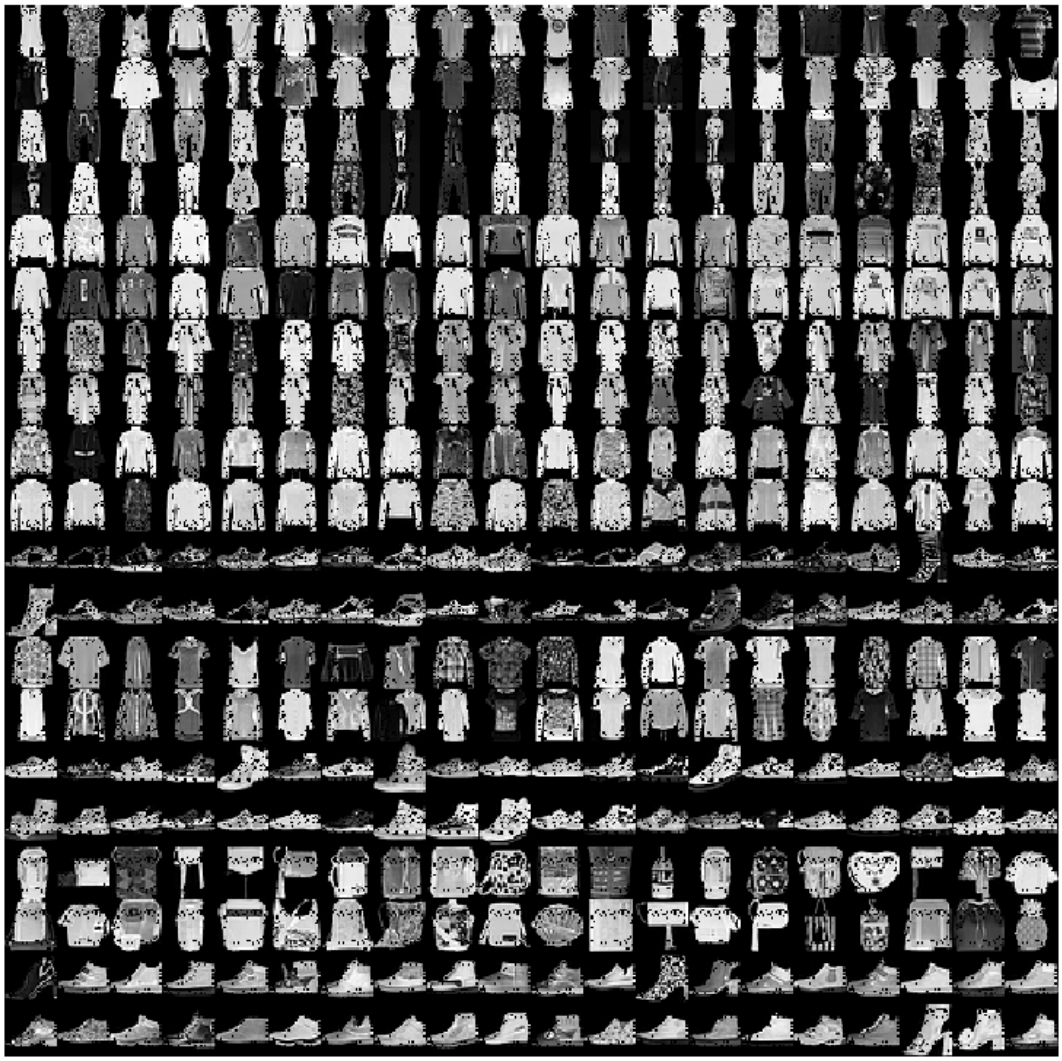


Figure 11: Bootstrapped positive samples of Fashion-MNIST.

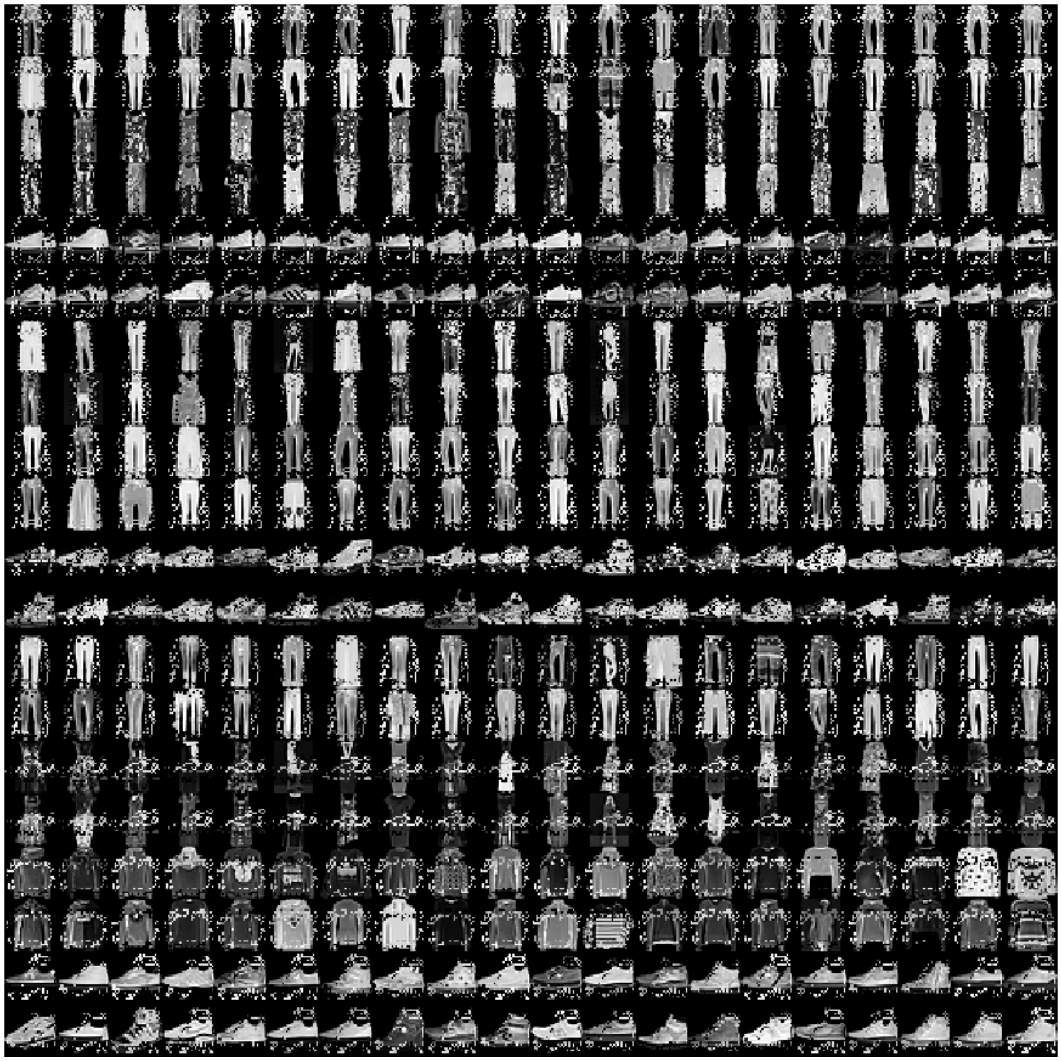


Figure 12: Bootstrapped negative samples of Fashion-MNIST.

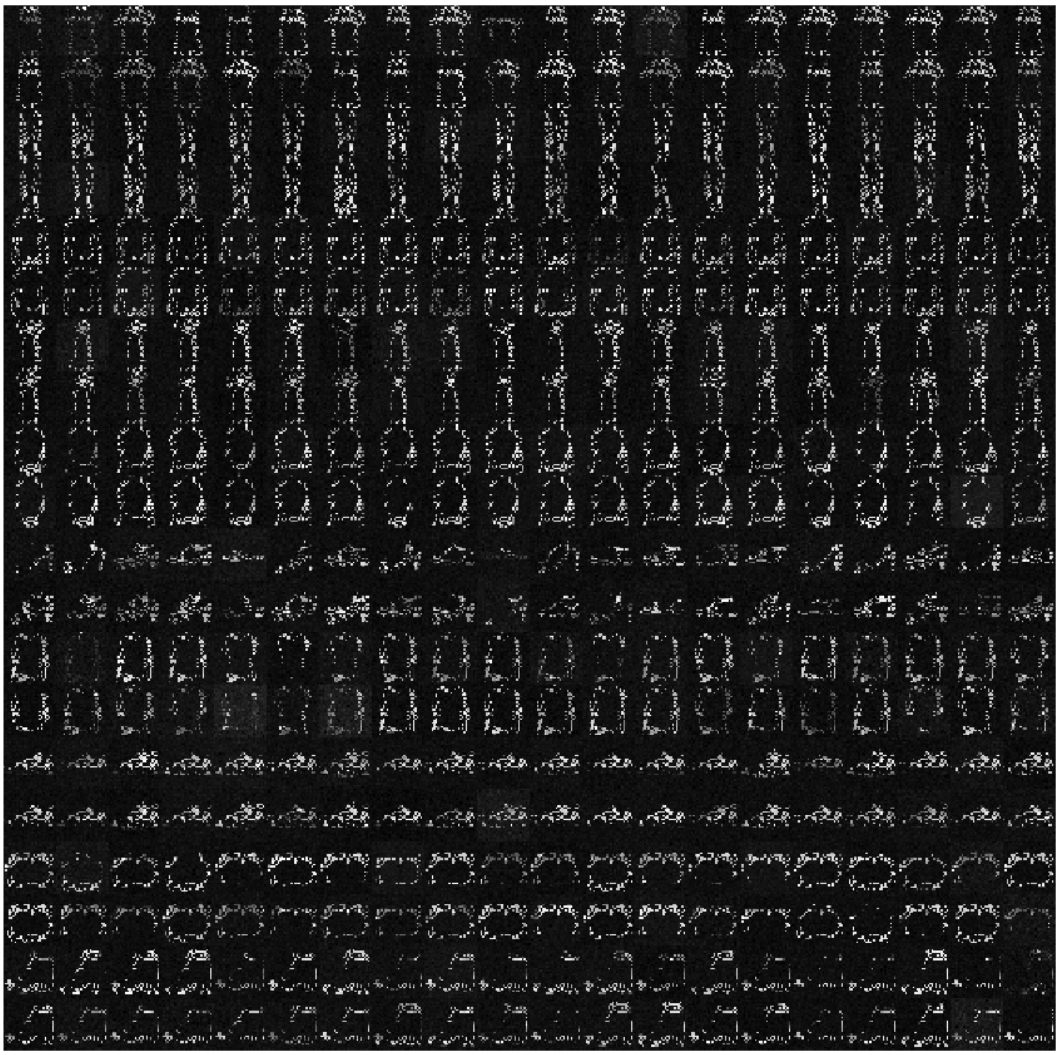


Figure 13: New testing cases of of Fashion-MNIST.