# A Framework to Learn with Interpretation

**Jayneel Parekh** [1], **Pavlo Mozharovskyi** [1], **Florence d'Alché-Buc** [1]
[1] LTCI, Telecom Paris, Institut Polytechnique de Paris
{jayneel.parekh, pavlo.mozharovskyi, florence.dalche}@telecom-paris.fr

## Abstract

With increasingly widespread use of deep neural networks in critical decision-making applications, interpretability of these models is becoming imperative. We consider the problem of jointly learning a predictive model and its associated interpretation model. The task of the interpreter is to provide both local and global interpretability about the predictive model in terms of human-understandable high level attribute functions, without any loss of accuracy. This is achieved by a dedicated architecture and well chosen regularization penalties. We seek for a small-size dictionary of attribute functions that take as inputs the outputs of selected hidden layers and whose outputs feed a linear classifier. We impose a high level of conciseness by constraining the activation of a very few attributes for a given input with a real-entropy-based criterion while enforcing fidelity to both inputs and outputs of the predictive model. A major advantage of simultaneous learning is that the predictive neural network benefits from the interpretability constraint as well. We also develop a more detailed pipeline based on some common and novel simple tools to develop understanding about the learnt features. We show on two datasets, MNIST and QuickDraw, their relevance for both global and local interpretability.

## Introduction

Owing to the high competitive performance, machine learning methods and specially deep neural networks have recently made major inroads in classification and decision-making applications. However their deployment in critical environments such as applications in defence, law, healthcare or any domain that has a high impact on human lives has strongly induced the need to understand the "rationale" behind their decisions. This has even led to impose interpretability and explainability of these systems as part of legal and human rights (see General Data Protection Regulation[1]). It is important to stress that explainability as well as interpretability are often used as synonyms in the literature, referring to the ability to provide human-understandable insights on the decision process. Throughout this paper, we opt for interpretability as in (Doshi-Velez and Kim 2017) and leave the term explainability for the ability to provide logical explanations or causal reasoning, both requiring more sophisticated frameworks (Dubois and Prade 2014; Guidotti et al. 2019). To address the long-standing challenge of interpreting black-box models such as deep neural networks (Samek et al. 2019; Beaudouin et al. 2020; Barredo Arrieta et al. 2020), two main approaches have been developed in literature: post-hoc approaches and "by design methods".

Post-hoc approaches like saliency maps (Springenberg et al. 2014; Selvaraju et al. 2017) and perturbation based methods (Lundberg and Lee 2017) address the interpretability issue by considering a trained network and analyze a posteriori its behavior by performing attribution over input features or building a local proxy model, generally linear, in the vicinity of a datapoint. While these approaches have undoubtedly increased the understanding of a network's decision, they fail to capture a global picture and are criticized for not explaining the predictor function itself.

This drawback is alleviated by another line of research, "interpretability by design" (Al-Shedivat, Dubey, and Xing 2017; Melis and Jaakkola 2018), which aims at integrating the interpretability objective in the learning process. This long-standing approach modifies the structure of predictor function itself to make it equivalent to a set of logical rules and/or adds to the loss function regularizing penalties to enforce interpretability. Although the literature has not reached a consensus on these properties, one can mention consistency, completeness, stability and sparsity. However, in this attractive family of approaches, the interpretability is often paid at the expense of accuracy. The goal of this work is to propose and study a novel framework to learning interpretable networks while overcoming the trade-off between accuracy and interpretability. Our starting point rely upon two notions of interpretability: for a given input, local interpretability is the ability of a model to provide a small subset of high level attributes, whose simultaneous activation leads to the model's prediction. Global interpretability corresponds here to the whole picture, i.e. the different sets of attributes that are associated to class prediction. In our novel approach called FLINT (Framework for Learning Interpretable Networks), we consider a pair of models, a predictor network, and an interpreter network, whose architectures are closely linked, and that are jointly learned. There is no strong assumption on the predictor network, except that it is a deep architecture and thus it is able to learn intermediate representations of the input. Any deep neural network

Preprint, paper under review
[1] https://gdpr-info.eu/

is therefore eligible. The interpreter architecture which can be considered as a global proxy of the predictor function exhibits two main features. First it takes as input the outputs of several hidden layers of the predictor network which enforces by design a closeness to the predictor network. Second it relies on a decomposition of its prediction on a dictionary of high level attribute functions. Note that this kind of generalized linear model has already been used in other works but neither taking the outputs of several hidden layer nor being simultaneously trained with the predictor network. This task sharing is beneficial for both networks as we show it in experiments: the predictor network even slightly improves its accuracy while the interpreter network builds on the relevant hidden representations learned by the predictor. A key feature of this approach is to design a loss function that impose a limited number of attribute functions as well as conciseness and diversity among the activation of these attributes for a given input. The paper is structured as follows. We first provide a brief overview of key features of related works, then we introduce the whole framework. Experiments conducted on two image classification datasets, MNIST and QuickDraw, are discussed before drawing conclusion and perspectives.

## Related Works

**Interpretability as part of learning** Most works for interpretability focus on producing*a posteriori* (or *post-hoc*) explanations. They often consider the model as a black-box (Lundberg and Lee 2017; Lakkaraju, Arsov, and Bastani 2020; Lakkaraju et al. 2019; Bang et al. 2019) or in the case of deep neural networks, work with gradients to generate saliency maps for a given input (Simonyan, Vedaldi, and Zisserman 2013; Smilkov et al. 2017; Selvaraju et al. 2017; Montavon, Samek, and Müller 2018). There are a few works that modify either the structure of the model (Al-Shedivat, Dubey, and Xing 2017; Angelov and Soares 2020), the losses (Lee et al. 2019), or both (Zhang, Nian Wu, and Zhu 2018; Melis and Jaakkola 2018) to incorporate interpretability in the learnt model. However all these approaches offer local interpretations, with the exception of LIME (Ribeiro, Singh, and Guestrin 2016) and SHAP (Lundberg and Lee 2017). Recently proposed neural additive models (Agarwal et al. 2020) are both, globally and locally interpretable. However, their approach is currently only suitable for tabular data, and not for more complex input like images.

**Interpretability via human understandable features** Traditional methods for producing explanations generally perform attribution over input features. They have been questioned for their robustness to simple transformations to input such as constant shifts (Kindermans et al. 2017). This has led to interest in use of human-interpretable concepts for interpretations. TCAV (Kim et al. 2017) propose to utilize human-annotated examples to represent concepts in terms of activations of a pre-trained neural network, termed as *concept activation vector* (CAV). The sensitivity of these concepts for prediction of each class is estimated to offer a global explanation. ACE (Ghorbani et al. 2019) attempts to automate the human-annotation process by super-pixel segmentation and clustering. ConceptSHAP (Yeh et al. 2019) further builds by introducing the idea of "completeness" or importance of concepts in this framework. Self explaining neural networks (SENN) (Melis and Jaakkola 2018) presented a generalized linear model. The linear structure is to emphasize interpretability. However to not lose out on performance the coefficients are modelled as a function of input. They are modelled using much heavier neural networks compared to the features. SENN imposes a gradient-based penalty to learn coefficients stably and constraints to learn human understandable features.

The CAV-based approaches strongly differ from ours in the context of the problem as they produce post-hoc interpretations. Moreover, their candidate concepts are externally generated, either by human-supervision, or pre-processing algorithms, and not from the network itself. The pre-processing steps also limit the applicability on types of problems. In case of SENN, although the coefficients are learnt so as to behave stably, they are still output of an uninterpretable system. This non-interpretable modelling leads to major differences with our framework. To retain interpretability, we don't model the coefficients as functions of input but rather as a linear classifier. To not trade-off accuracy we allow the classifier to be modelled as unrestrained neural network and learn an interpreter separately. Moreover, we build an extensive pipeline for better understanding of the learnt features for both local interpretations and global interpretations.

## Our framework: FLINT

We describe our framework for a multi-class classification problem with $C$ classes. Denote $\mathcal{X} = \mathbb{R}^d$ the input space and $\mathcal{Y}$, the output space, the set of $C$ one-hot encoding vectors of dimension $C$. A deep neural network with $l+1$ layers of respective dimension $d_1, \ldots, d_{l+1}$ is a mapping $f : \mathcal{X} \to \mathcal{Y}$ such that: $f = f_{l+1} \circ f_l \circ ... \circ f_1$ where $f_k : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}, k = 1, ..., l+1$ is the function implemented by layer $k$. For any $x \in \mathcal{X}$, the output $f_k(x)$ of an intermediate layer $k$ in $f(x)$ is denoted $x^k$ while its dimension is called $d_k$. We call $\mathcal{F}_l$ the space of deep neural networks with $l+1$ layers. Alternatively, a network $f$ in $\mathcal{F}_l$ is completely identified by its generic parameter $\theta_f \in \Theta$. We describe our framework for a multi-class classification problem with $C = d_{l+1}$ classes. We assume that the training set $\mathcal{S} = \{(x_i, y_i)_{i=1}^N\}$ is an i.i.d. sample from a fixed but unknown joint probability distribution $P(X, Y)$ over $\mathcal{X} \times \mathcal{Y}$.

### Learning with interpretation

Given the training set $\mathcal{S}$, our goal is to jointly learn a pair of models, a prediction network $f$ and an interpretation network $g$. We make no special assumption on $f$: $f$ can be chosen as the state-of-the-art architecture for the multi-class classification task at hand. The role of the interpreter is to provide both local and global interpretability about $f$ in terms of high level attribute functions, without any loss of accuracy for $f$. The interpreter model $g$ takes as inputs the outputs of several hidden layers of $f$ to benefit from its representation learning ability. We introduce *Supervised Inter-*

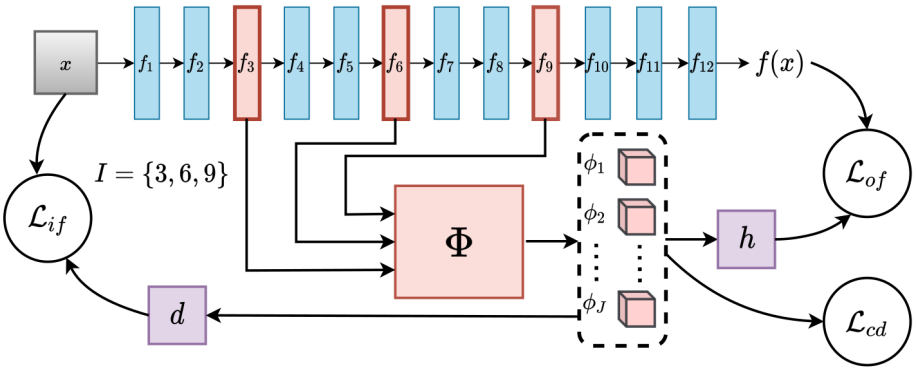



Figure 1: General scheme of FLINT. The predictor $f$ is a deep neural network with multiple layers. The interpreter $g = h \circ \Phi$. $\Phi$ takes as input intermediate outputs of $f$, and outputs human-interpretable attributes and $h$ is a linear classifier

*pretable Learning* in the context of empirical risk minimization as the following joint optimization problem:

$$\arg\min_{f,g} \mathcal{L}_{pred}(f, \mathcal{S}) + \mathcal{L}_{int}(f, g, \mathcal{S}),$$

where $f$ is searched over $\mathcal{F}_l$ and $g$ over the class of interpreter networks of $f$ that we define below, $\mathcal{L}_{pred}(f, \mathcal{S})$ denotes a loss term related to prediction and $\mathcal{L}_{int}(f, g, \mathcal{S})$ is a loss term related to interpretability.

**Interpretation and interpreter network** Before defining each of the loss terms, we emphasize that we now consider three possible uses of this pair of models. The first one is the usual task of prediction, i.e. computing $\hat{y} = f(x)$. The second one, called here *local interpretation*, consists in providing insights about $f(x)$ by extracting information from $g(x)$. This should be done only if $g(x) = f(x)$. Indeed, in case $g(x)$ and $f(x)$ disagree, this raises the questions of i) trusting the prediction $f(x)$, ii) augmenting the capacity of $g$ and ii) exploiting however the information brought by function $g$. We discuss this issue in the supplements. The third use of our pair of models is to provide a global interpretation of the way each class is predicted by $f$.

In this work, a *local interpretation* of $f(x)$ is defined as a small set of high level attribute functions which are highly activated when the prediction $\hat{y} = g(x) = f(x)$ is computed. A global interpretation is a description of each class in terms of a small subset of high level attribute functions built from the predictor network. Moreover, a given attribute function captures a property of the input data that may be useful to predict different classes.

Given this definition, an interpreter network $g$ provides a prediction by linearly combining several attribute functions $\phi_j, j = 1, \dots J$ that take as input, the outputs of several selected hidden layers (see Fig. 2). Let us call $\mathcal{I} = \{i_1, i_2, ..., i_T\} \subset \{1, \dots, l+1\}$ the set of indexes specifying the layers to be accessed. We define $D = \sum_{t=1}^{T} d_{i_t}$. Typically these layers are selected from the latter layers of the network $f$. The concatenated vector of all intermediate outputs for an input sample $x$ is then denoted as $x^{\mathcal{I}} = f_{\mathcal{I}}(x) \in \mathbb{R}^D$. The dictionary of attribute functions $\phi_j : \mathbb{R}^D \to \mathbb{R}^+, j = 1, ..., J$ is modelled by a shallow neural network, consisting only of 2-3 layers. The vector consisting of all attributes is denoted as $\Phi(x^{\mathcal{I}}) = [\phi_1(x^{\mathcal{I}}), \phi_2(x^{\mathcal{I}}), ..., \phi_j(x^{\mathcal{I}})] \in \mathbb{R}^J$. Eventually, the interpreter network computes the following function:

$$\forall x \in \mathcal{X}, g(x) = h \circ \Phi(x^{\mathcal{I}}),$$

where $h : \mathbb{R}^J \to \mathbb{R}^C$ is a linear model composed with a "softmax" layer. For sake of simplicity, we denote $\Theta_g = (\theta_\Phi, \theta_h)$ the parameters of this model. Note that other candidates such as decision trees could be used instead of the linear classifier $h$.

## Imposing interpretability properties

We adopt here an axiomatic view of interpretability. To overcome the difficulty of defining what is a human-understandable attribute function, we list a number of properties desirable for an interpreter network and enforce them via various penalty terms additionally to the architecture choices. Although this kind of approach is shared by several works in Explainable AI (Carvalho, Pereira, and Cardoso 2019), there is no consensus on these properties. We propose here a minimal set of penalties which are relevant for the proposed architecture and that is sufficient to provide relevant attribute functions.

**Fidelity to Output.** Since the attributes are expected to provide interpretability to the network $f$'s prediction, the output of the interpreter $g(x)$ should be "close" to the output $f(x)$ for any $x$. This can be imposed through a cross-entropy loss:

$$\mathcal{L}_{of}(f, g, \mathcal{S}) = -\sum_{x \in \mathcal{S}} h(\Phi(x^{\mathcal{I}})^T log(f(x))$$

**Conciseness and Diversity.** For any given sample $x$, we wish to get a very small number of attribute functions needed to encode relevant information about it. This property of "conciseness" should help in making the interpretations easier to understand. However, to encourage better use of available attributes we also expect use of multiple attributes across many randomly selected samples.We refer to

this property as *diversity*. To enforce these conditions we rely on the notion of entropy defined for real vectors as proposed for instance by (Jain et al. 2017). For a real-valued vector $v$, the entropy is defined as $\mathcal{E}(v) = -\sum_i p_i log(p_i)$, $p_i = exp(v_i)/(\sum_i exp(v_i))$.

Conciseness is promoted by minimizing $\mathcal{E}(\Phi(x^{\mathcal{I}}))$ and diversity is promoted by maximizing entropy of average attribute vector over a mini-batch. Since entropy-based losses have an inherent normalization, they do not constrain the magnitude of the attributes. This often leads to poor optimization. Thus, we also minimize the $\ell_1$ norm $\|\Phi(x^{\mathcal{I}})\|_1$ to avoid it. Note that $\ell_1$-regularization is a common tool to encourage sparsity and thus conciseness, however we show in the experiments that entropy provides a more effective way.

$$\bar{\Phi}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|}\sum_{x\in\mathcal{S}} \Phi(x^{\mathcal{I}})$$

$$\mathcal{L}_{cd}(\Phi,\mathcal{S}) = \zeta\Big[-\mu\mathcal{E}(\bar{\Phi}_{\mathcal{S}}) + \sum_{x\in\mathcal{S}} \mathcal{E}(\Phi(x^{\mathcal{I}}))\Big] + \sum_{x\in\mathcal{S}} \eta\|\Phi(x^{\mathcal{I}})\|_1$$

**Fidelity to Input.** To encourage encoding of high-level meaningful patterns, $\Phi(x^{\mathcal{I}})$ should encode relevant information about the input $x$. Following SENN (Melis and Jaakkola 2018), we impose this using a decoder network $d : \mathbb{R}^D \to \mathcal{X}$ that takes as input the set of attributes $\Phi(x^{\mathcal{I}})$ and reconstructs $x$.

$$\mathcal{L}_{if}(d,\Phi,f,\mathcal{S}) = \sum_{x\in\mathcal{S}} (d(\Phi(x^{\mathcal{I}})) - x)^2$$

## Learning Algorithm

Given the proposed loss terms, the loss for interpretability writes as follows:

$$\mathcal{L}_{int}(f,\Phi,h,d,\mathcal{S}) = \beta\mathcal{L}_{of}(f,\Phi,h,\mathcal{S}) + \gamma\mathcal{L}_{if}(\Phi,h,d,\mathcal{S}) + \delta\mathcal{L}_{cd}(\Phi,\mathcal{S})$$

For $\mathcal{L}_{pred}$, cross-entropy loss can be used as well. Regularization can also be performed through different mechanisms.

Let us denote $\Theta = (\theta_f, \theta_d, \theta_\Phi, \theta_h)$ the parameters of these networks. Learning the parametric models $f$, $\Phi$, $h$ and $d$ boils down to learn their parameter. In practice, introducing all the losses at once often leads to very poor optimization. Thus for both the datasets we follow the procedure described in algorithm 1. We train the networks with $\mathcal{L}_{pred}, \mathcal{L}_{if}$ for the first two epochs and gain a reasonable level of accuracy. From the third epoch we introduce $\mathcal{L}_{of}$ and $\ell_1$-regularization and from the fourth epoch we introduce the entropy losses.

## Interpretation Phase

We now present how to generate a global interpretation for $f(.)$ or local interpretation for any $x$ with prediction $f(x)$.

**Global interpretability.** We create a pipeline to extract class-attribute pairs important for interpretation. We statistically estimate the relative importance of an attribute $\phi_j$ for class $c$, denoted by $r_{j,c}$. Although this could be directly done by analyzing parameters of $h$, statistically estimating it helps

**Algorithm 1** Learning Algorithm for FLINT

Input: $\mathcal{S}$ & hyperparameter: $\beta_0, \gamma_0, \delta_0, \mu_0, \zeta_0, \eta_0$ & number of batches $B$, $s$: size of a batch.
$\beta = 0, \gamma = \gamma_0, \delta = 0$
Random initialization of parameter $\Theta$
**for** $m = 0$ to $N_{epoch} - 1$ **do**
  **if** m == 2 **then**
    $\beta = \beta_0, \delta = \delta_0, \eta = \eta_0, \zeta = 0$ ▷ No entropy
  **if** m == 3 **then**
    $\mu = \mu_0, \zeta = \zeta_0$
  **for** $b = 1$ to $B$ **do**
    Pick up a mini-batch of size $s$
    Compute $\mathcal{L}_{pred}, \mathcal{L}_{of}, \mathcal{L}_{if}, \mathcal{L}_{cd}$ for mini-batch
    $\mathcal{L} = \mathcal{L}_{pred} + \beta\mathcal{L}_{of} + \gamma\mathcal{L}_{if} + \delta\mathcal{L}_{cd}$
    $\theta_f, \theta_\Phi, \theta_h, \theta_d \leftarrow$ Backprop($\mathcal{L}$)
**return** $\Theta$

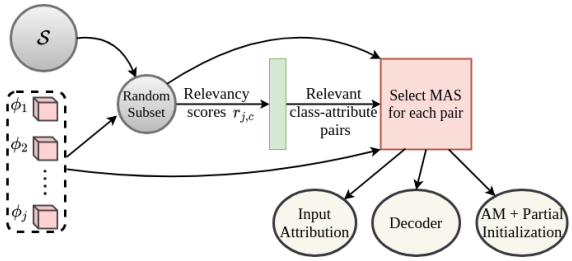



Figure 2: Flow for generating global interpretations

in selecting the prominently activating attributes and class-attribute pairs commonly occurring in interpretations. We propose use of multiple tools to gain understanding about patterns detected by attributes in the extracted pairs.

**Selecting relevant class-attribute pairs** For a sample $x$ with predicted class $\hat{c}$ (by the interpreter, $h(\Phi(x^{\mathcal{I}}))$), we define the total contribution of attribute $j$ as $\alpha_{j,\hat{c},x} = \phi_j(x^{\mathcal{I}}).w_{j,\hat{c}}$, where $w_{j,\hat{c}}$ are weights of linear classifier $h$. The importance of attribute $j$, for predicting class $\hat{c}$, for sample $x$ is, $r_{j,\hat{c},x} = \frac{\alpha_{j,\hat{c},x}}{max_i|\alpha_{i,\hat{c},x}|}$. To estimate $r_{j,c}$ for a given class-attribute pair $(j,c)$, we compute mean of $r_{j,\hat{c},x}$ for samples $x$ where predicted class $\hat{c} = c$. That is, $r_{j,c} = \sum_{\{x\in\mathcal{S}_{rnd}|\hat{c}=c\}} r_{j,\hat{c},x}$ ($\mathcal{S}_{rnd}$ is random subset of the training set). To select relevant class-attribute pairs, we simply threshold $r_{j,c}$ for each $(j,c)$. For each such selected pair we analyze the attribute's maximum activating samples (MAS) from the class using the tools discussed below.

**Analysis tools for interpretability** To understand patterns encoded by any attribute on the selected MAS for a class-attribute pair, we use the following tools:

- *Input attribution*: This is a natural choice to understand an attribute's action for a sample. There are a variety of algorithms that can be employed ranging from black-box local explainers to saliency maps. While these maps are less noisy and easily obtained, they do not enhance encoded patterns and thus the underlying patterns may not

be easily distinguishable from the other parts of input.

- *Decoder*: As mentioned earlier, we also train an autoencoder via a decoder $d$ that uses the attributes as input. Thus, for an attribute $j$ and input $x$, we can compare the reconstructed samples $d(\Phi(x^{\mathcal{I}}))$ and $d(\Phi(x^{\mathcal{I}})\backslash j)$ where $\Phi(x^{\mathcal{I}})\backslash j$ denotes attribute vector with $\phi_j(x^{\mathcal{I}}) = 0$, i.e., removing the effect of attribute $j$.
- *Activation maximization and partial initialization (AM+PI)*: To focus more on enhancing a pattern detected by an attribute, synthesizing appropriate input via optimization is a common and popular idea, referred to as activation maximization (Mahendran and Vedaldi 2016). In our case this optimization problem for an attribute $j$ is:

$$\arg\max_x \lambda_\phi \phi_j(x^{\mathcal{I}}) - \lambda_{tv}\text{TV}(x) - \lambda_{bo}\text{Bo}(x)$$

where $\text{TV}(x)$ denotes total variation of $x$ and $\text{Bo}(x)$ promotes boundedness of $x$ in a range. For details, refer to (Mahendran and Vedaldi 2016). Empirically, we found that for very deep predictors, this optimization with random initialization often failed (attribute failed to activate) or lead to noisy results on our datasets. Thus, we initialize the optimization procedure by low-intensity version of selected input sample. This makes the optimization problem easier with the attribute's detected pattern weakly present in the input. This also allows the optimization to "fill out" the input on its own with the desired pattern of attribute. For the output we obtain a map consisting of the attribute's detected pattern adapted to the particular input. We treat this as our primary tool for understanding patterns detected by an attribute.

**Local interpretability** To interpret the prediction $f(x)$ for any sample $x$ one only needs to extract the importance score $r_{j,\hat{c},x}$ ($\hat{c} = h(\Phi(x^{\mathcal{I}}))_{argmax}$), for each attribute, and then put these with the interpretations generated for each attribute. The tools discussed above can be used to gain more precise understanding about what each attribute is detecting. One can modify $\hat{c}$ and repeat the same analysis to gain understanding about relevance of attributes to a different class.

## Numerical Experiments and discussion

### Datasets and Networks

We consider 2 datasets for experiments, MNIST (Lecun et al. 1998) for digit recognition and subset of QuickDraw dataset for sketch recognition (Ha and Eck 2018). Additional results on FashionMNIST (Xiao, Rasul, and Vollgraf 2017) are available in supplementary material. In the case of QuickDraw, we select 10000 random images from each of 10 classes: 'Ant', 'Apple', 'Banana', 'Carrot', 'Cat', 'Cow', 'Dog', 'Frog', 'Grapes', 'Lion'. We randomly divide each class into 8000 training and 2000 test images. Complete data is provided in the supplementary material.

Our experiments include 2 kinds of architectures for predictor $f$: (i) LeNet-based (LeCun et al. 2015) network, and (ii) Adapted ResNet18 (He et al. 2016) (no batch normalization layers). $\Phi$ consists of 1–2 conv. layers followed by pooling and a FC layer. For LeNet based network, we fix $\mathcal{I}$ as output of $2^{nd}$ conv. layer. For the ResNet18-based network, we fix $\mathcal{I}$ as output of the $3^{rd}$ conv. block (*13th* conv. layer) and middle of the $4^{th}$ conv. block (*15th* conv. layer). Precise architecture details are given in the supplementary.

**Optimization** All the models are trained for 12 epochs. We use Adam (Kingma and Ba 2014) as the optimizer with fixed learning rate $0.0001$ and train on a single NVIDIA-Tesla P100 GPU. Quantitative metrics on QuickDraw with ResNet are averaged across 3 runs for each set of parameters. Implementations are done using PyTorch (Paszke et al. 2017).

**Hyperparameter tuning** For our experiments we set the number of attributes to $J = 25, 24$ for MNIST and QuickDraw, respectively. For MNIST with LeNet, we set $\zeta = 1, \mu = 1, \eta = 0.5, \delta = 0.2$, and for QuickDraw with ResNet, to emphasize conciseness less and diversity more we set $\zeta = 1, \mu = 2, \eta = 3, \delta = 0.1$. We set $\delta = 0.1$ for experiments on QuickDraw and $\delta = 0.2$ on MNIST. $\beta = 0.1$ is employed for QuickDraw and $\beta = 0.5$ is employed for MNIST. It's slightly more tedious to tune $\gamma$. $\gamma$ is varied between 0.8 to 20. We tune it so that the average value of $\mathcal{L}_{if}$ on $\mathcal{S}$ at least halves by the last epoch of training. $\gamma$ is set to $0.8$ for MNIST and $5.0$ for QuickDraw.

**Choices for interpretation phase** For a random subset $\mathcal{S}_{rnd}$ consisting of $1000$ samples from $\mathcal{S}$, we select class-attribute pairs which have $r_{j,c} > 0.1$ and use gradient as attribution method for LeNet based network and Guided Backpropagation (Springenberg et al. 2014) for ResNet based network. We fix parameters for AM+PI for all our experiments as $\lambda_\phi = 2, \lambda_{tv} = 6, \lambda_{bo} = 10$ and for each sample $x$ to be analyzed, we analyze input for this optimization as $0.1x$. For optimization, we use Adam with learning rate $0.05$ for $300$ iterations, halving learning rate every $50$ iterations.

### Evaluating the trained models

We evaluate our model on three quantitative metrics and qualitatively analyze the learnt attributes. The metrics are:

- **Accuracy loss**: To quantify effect of jointly learning the interpreter on the performance of $f$, we also train it with $\beta, \gamma, \delta = 0$ (i.e only with $\mathcal{L}_{acc}$) and compare the difference in accuracy on test set.
- **Fidelity of interpreter**: We quantify how well $g$ interprets $f$ by computing number of samples where prediction of $g$ is same as $f$.
- **Conciseness of interpretations**: We propose the following metric to measure the conciseness of an interpretation based on high-level features: For a given sample $x$, let the predicted class of interpreter $g$ be $\hat{c}$ and $r_{j,\hat{c},x}$ the relevance for attribute $j$. We calculate conciseness for sample $x$, $\text{CNS}_{g,x}$, as number of attributes with $r_{j,\hat{c},x}$ greater than a threshold $1/\tau, \tau > 1$, i.e. $\text{CNS}_{g,x} = |\{j : r_{j,\hat{c},x} > 1/\tau\}|$. for different thresholds $\tau$, We compute the mean of $\text{CNS}_{g,x}$ over test data to estimate conciseness of $g$, $\text{CNS}_g$.

### Quantitative analysis

Tab. 1 indicates the accuracy of various networks related to FLINT. Training $f$ within FLINT does not result in any

| Dataset | Architecture | System | Accuracy (%)↑ |
|---|---|---|---|
| MNIST | LeNet | BASE-$f$ | 99.0 |
| | | FLINT-$f$ | 99.1 |
| | | FLINT-$g$ | 98.4 |
| QuickDraw | ResNet | BASE-$f$ | 85.2 |
| | | FLINT-$f$ | 85.4 |
| | | FLINT-$g$ | 85.3 |

Table 1: Accuracy on different datasets. BASE-$f$ is system trained with just accuracy loss. FLINT-$f$, FLINT-$g$ denote the predictor and interpreter trained in our framework.

| Dataset | Architecture | Fidelity (%)↑ |
|---|---|---|
| MNIST | LeNet | 98.8 |
| QuickDraw | ResNet | 90.8 |

Table 2: Fidelity results for FLINT

accuracy loss on both datasets, which can be primarily attributed to the design of the framework: Although all the additional losses regularize the intermediate layers of $f$ (their gradient propagates to all layers before and including the most advanced layer in $\mathcal{I}$), these constraints are not directly placed on the layers of $f$ (specially ones close to output), but propagate through $\Phi$. The flexibility in choosing the set of intermediate layers to be used for attribute function is also a useful possibility in this regard.

Tab. 2 tabulates the fidelity of $g$ to output of $f$. We achieve very high fidelity on MNIST, which is mainly because $g$ accesses intermediate layers of $f$, but also since learning objective encourages $g$ to match output of $f$. The fidelity on QuickDraw is also reasonably high, and the lacking $9.2\%$ can be explained by two main factors, see the supplementary material for illustration. First, e.g., in Apple, a very few wrongly attributed by $g$ observations are outliers. Second, a subclass (e.g. of Cat) is too small and diverse to generate intuitive features, while $f$ fits to it in an "unexplainable" (by the features) way. Further, Tab. 3 captures the effect of entropy and $\ell_1$-regularization strength on fidelity of $g$ for QuickDraw illustrating the (relatively small) price in terms of fidelity to pay for the sparsity when adding entropy.

To evaluate the conciseness of FLINT-$g$ (also on QuickDraw), we plot the average number of relevant attributes for different values of $\eta$ and $\xi$ as a function of threshold.

The figures confirm that using the entropy-based loss is a more effective way of inducing conciseness of explanations compared to using just $\ell_1$-regularization, with the difference being close to use of 1 attribute less when entropy losses are employed. On MNIST, the difference is even more pronounced because of higher weight for entropy losses $\zeta$, weaker $\ell_1$-regularization $\eta$, and more focus on conciseness during training (lower $\mu$).

| | $\eta=1$ | $\eta=2$ | $\eta=3$ | $\eta=5$ |
|---|---|---|---|---|
| $\zeta=0$ | 92.7 | 90.4 | 91.2 | 84.2 |
| $\zeta=1$ | 91.2 | 90.7 | 90.8 | 82.9 |

Table 3: Fidelity variation for $\eta$ and $\zeta$. All other parameters fixed

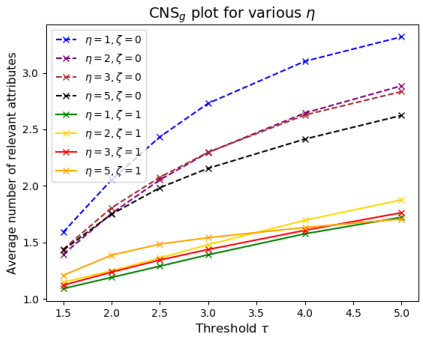



Figure 3: Effect of entropy losses on conciseness of ResNet with QuickDraw for various $\ell_1$-regularization levels.

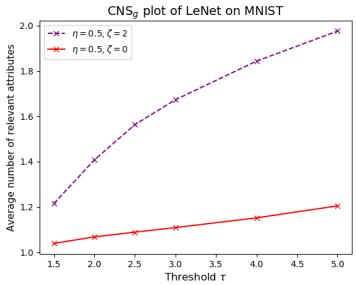



Figure 4: Conciseness of LeNet with MNIST

## Qualitative analysis

**Global interpretation.** Fig. 5 depicts part of the global interpretability picture with various class-attribute pairs detected for QuickDraw. We select attributes most relevant for two of the best (Apple and Banana) and worst (Dog and Cow) classes in terms of fidelity and me of these attributes are also relevant for other classes, we include the pairs for those classes as well. We show 8 out of 24 class-attribute pairs in Fig. 5. For each class-attribute pair we analyse AM+PI output for 3 MAS. Comprehensive illustration on all class-attribute pairs can be found in the supplementary material. As expected, there are attributes exclusive to a class, while others are relevant for multiple classes. $\phi_3$ (relevant for Apple only) clearly detects large circular shape, while $\phi_{16}$ "draws" multiple half-strokes of a banana. Multi-class attributes $\phi_{22}, \phi_5, \phi_{15}$, though more complex, still consistently detect similar patterns. This significantly contributes to their understandability. $\phi_{22}$ shows strong activation for blotted textures, often drawn on stomach of a cow, or as dense blobs of grapes. $\phi_{15}$ (most relevant for Cow, Dog, & Ant) primarily detects leg-like vertical strokes. However it also detects strokes related to head and body, which can correspond to tentacles in ant and stomach for Dogs/Cows.

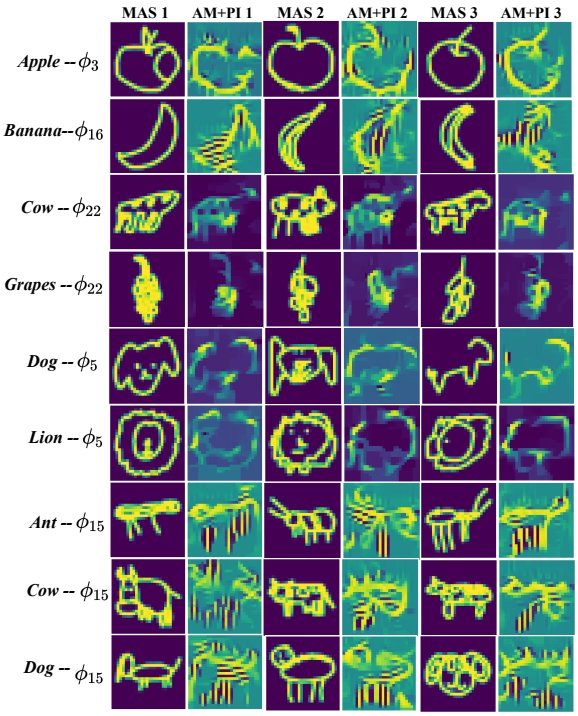



Figure 5: Attributes learnt on QuickDraw with ResNet. Rows correspond to a class-attribute pairs. 3 maximum activating samples are analyzed with their AM+PI outputs.

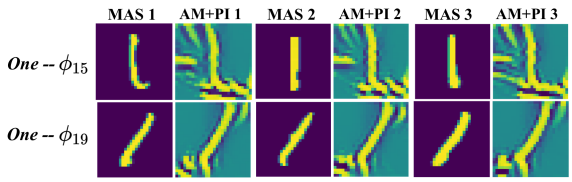



Figure 6: Sample attributes learnt on MNIST with LeNet. Each row corresponds to a class-attribute pair. Three MAS are analyzed with their AM+PI outputs.

Likewise we show sample results of global interpretability for MNIST in Fig. 6. Complete sets of figures are in the supplementary. For MNIST, most attributes are class-exclusive. This makes them straightforward to understand. An interesting observation about the learnt attributes is that for classes with multiple relevant attributes, like class '1', each attribute activates to specific types of samples for that class. E.g. $\phi_{19}$ detects 1's with diagonal strokes and $\phi_{15}$ detects 1's with vertical strokes. This again contributes to the understandability of the attributes. This is less observed for attributes learnt for QuickDraw with ResNet, most likely because of variety (and complexity) of samples.

**Local interpretation.** Fig. 7 displays the local interpretations generated for test samples of QuickDraw. $f$ and $g$ both predict the true class in all the cases. We show the top 3 relevant attributes to the prediction and their corresponding AM+PI images for QuickDraw. For the first sample, from

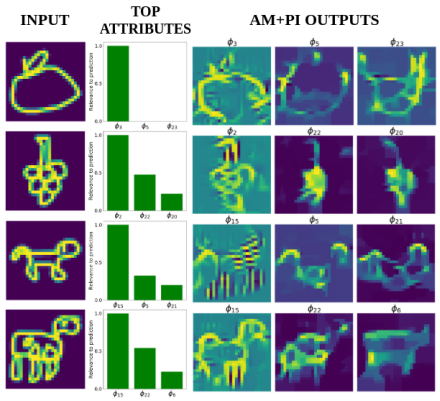



Figure 7: Local interpretations for test samples on QuickDraw. True labels for inputs are: Apple, Grapes, Dog, Cow.

class Apple, only one attribute $\phi_3$ is important. It detects pattern consistent with its global interpretation. Even $\phi_5$, though not relevant for prediction, detects very similar pattern as in global interpretations. Attributes $\phi_{15}, \phi_5, \phi_{22}$ also demonstrate the same consistency in patterns detected as expected from global interpretation outputs.

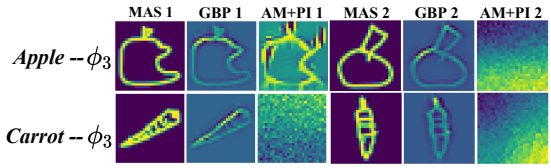



Figure 8: Sample attribute $\phi_3$ learnt without $\mathcal{L}_{if}$. GBP stands for Guided Backpropagation.

**Effect of autoencoder loss.** Although the effect of $\mathcal{L}_{of}, \mathcal{L}_{cd}$ can be objectively assessed to some extent, the effect of $\mathcal{L}_{if}$ can only be seen subjectively. If the model is trained with $\gamma = 0$, the attributes still demonstrate high overlap, nice conciseness. However, it becomes much harder to understand patterns encoded by them. For majority of attributes, MAS and the outputs of the analysis tools do not show any consistency of detected pattern. One such attribute, $\phi_3$, is depicted in Fig. 8. It is relevant for both Apple and Carrot, but it's hard to pin down a common pattern among all 4 images. Moreover, AM+PI fails to optimize on 3 out of 4 images. We provide figures for other attributes in the supplementary material. Such attributes are present even for the model trained with autoencoder, but are very few. We thus believe that autoencoder loss enforces a consistency in detected patterns for attributes. It does not necessarily enforce semantic meaningfulness in attributes, however it's still beneficial for improving their understandability.

## Conclusion

FLINT is a novel framework for learning a predictor network and its interpreter network with dedicated losses. A potential attractive use of our framework consists in retaining

only the interpreter model as the final prediction tool. Further works concern the investigation of this direction and the enforcement of additional constraints on attribute functions to encourage invariance and geometrical properties. Eventually nothing prevents to extend to other tasks in Machine Learning or other data than images.

## References


Agarwal, R.; Frosst, N.; Zhang, X.; Caruana, R.; and Hinton, G. 2020. Neural additive models: Interpretable machine learning with neural nets. *arXiv preprint arXiv:2004.13912* .

Al-Shedivat, M.; Dubey, A.; and Xing, E. 2017. Contextual explanation networks. *arXiv preprint arXiv:1705.10301* .

Angelov, P.; and Soares, E. 2020. Towards explainable deep neural networks (xDNN). *Neural Networks* 130: 185–194.

Bang, S.; Xie, P.; Wu, W.; and Xing, E. 2019. Explaining a black-box using Deep Variational Information Bottleneck Approach. *arXiv preprint arXiv:1902.06918* .

Barredo Arrieta, A.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; Garcia, S.; Gil-Lopez, S.; Molina, D.; Benjamins, R.; Chatila, R.; and Herrera, F. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58: 82 – 115.

Beaudouin, V.; Bloch, I.; Bounie, D.; Clémençon, S.; d'Alché-Buc, F.; Eagan, J.; Maxwell, W.; Mozharovskyi, P.; and Parekh, J. 2020. Flexible and Context-Specific AI Explainability: A Multidisciplinary Approach. *CoRR* abs/2003.07703.

Carvalho, D. V.; Pereira, E. M.; and Cardoso, J. S. 2019. Machine Learning Interpretability: A Survey on Methods and Metrics. *Electronics* 8(8).

Doshi-Velez, F.; and Kim, B. 2017. Interpretable Machine Learning. In *Proceedings of the Thirty-fourth International Conference on Machine Learning*.

Dubois, D.; and Prade, H. 2014. Possibilistic Logic-An Overview. *Computational logic* 9.

Ghorbani, A.; Wexler, J.; Zou, J. Y.; and Kim, B. 2019. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, 9277–9286.

Guidotti, R.; Monreale, A.; Giannotti, F.; Pedreschi, D.; Ruggieri, S.; and Turini, F. 2019. Factual and Counterfactual Explanations for Black Box Decision Making. *IEEE Intelligent Systems* .

Ha, D.; and Eck, D. 2018. A Neural Representation of Sketch Drawings. In *International Conference on Learning Representations*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

Jain, H.; Zepeda, J.; Pérez, P.; and Gribonval, R. 2017. Subic: A supervised, structured binary code for image search. In *Proceedings of the IEEE International Conference on Computer Vision*, 833–842.

Kim, B.; Wattenberg, M.; Gilmer, J.; Cai, C.; Wexler, J.; Viegas, F.; and Sayres, R. 2017. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *arXiv preprint arXiv:1711.11279* .

Kindermans, P.-J.; Hooker, S.; Adebayo, J.; Alber, M.; Schütt, K. T.; Dähne, S.; Erhan, D.; and Kim, B. 2017. The (Un)reliability of saliency methods.

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

Lakkaraju, H.; Arsov, N.; and Bastani, O. 2020. Robust and Stable Black Box Explanations. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Lakkaraju, H.; Kamar, E.; Caruana, R.; and Leskovec, J. 2019. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 131–138.

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

LeCun, Y.; et al. 2015. LeNet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet* 20(5): 14.

Lee, G.-H.; Jin, W.; Alvarez-Melis, D.; and Jaakkola, T. S. 2019. Functional Transparency for Structured Data: a Game-Theoretic Approach. *arXiv preprint arXiv:1902.09737* .

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 4765–4774.

Mahendran, A.; and Vedaldi, A. 2016. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision* 120(3): 233–255.

Melis, D. A.; and Jaakkola, T. 2018. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, 7775–7784.

Montavon, G.; Samek, W.; and Müller, K.-R. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* 73: 1–15.

Mosler, K. 2013. Depth Statistics. In Becker, C.; Fried, R.; and Kuhnt, S., eds., *Robustness and Complex Data Structures: Festschrift in Honour of Ursula Gather*, 17–34. Springer. Berlin.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch .

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

Samek, W.; Montavon, G.; Vedaldi, A.; Hansen, L. K.; and Müller, K., eds. 2019. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*. Springer.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, 618–626.

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

Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; and Wattenberg, M. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* .

Springenberg, J. T.; Dosovitskiy, A.; Brox, T.; and Riedmiller, M. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* .

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

Yeh, C.-K.; Kim, B.; Arik, S. O.; Li, C.-L.; Ravikumar, P.; and Pfister, T. 2019. On concept-based explanations in deep neural networks. *arXiv preprint arXiv:1910.07969* .

Zhang, Q.; Nian Wu, Y.; and Zhu, S.-C. 2018. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8827–8836.

Zuo, Y.; and Serfling, R. 2000. General notions of statistical depth function. *The Annals of Statistics* 28(2): 461–482.

# Supplementary material

## Architecture of ResNet, LeNet

Figs. 9 and 10 depict the architectures used for experiments with predictor architecture based on LeNet (LeCun et al. 2015) (on MNIST) and ResNet18 (on QuickDraw) (He et al. 2016) respectively.

## Experiments for number of attributes $J$

**Effect of $J$** We study the effect of choosing small values for number of attributes $J$ (keeping all other hyperparameters same). Tab. 4 tabulates the values of input fidelity loss $\mathcal{L}_{if}$, output fidelity loss $\mathcal{L}_{of}$ on the training data by the end of training for MNIST and the fidelity of $g$ to $f$ on MNIST test data for different $J$ values. Tab. 5 tabulates same values for QuickDraw. The two tables clearly show that using small $J$ can harm the autoencoder and the fidelity of interpreter. Moreover, the system packs more information in each attribute and this makes it hard to understand them, specially for very small $J$. This is illustrated in Figs. 11 and 12, which depict part of global interpretations generated on MNIST for $J = 4$ (all the parameters take default values). Fig. 11 shows global class-attribute relevances and Fig. 12 shows generated interpretation for a sample attribute $\phi_2$. It can be clearly seen that the attributes start encoding patterns for too many classes (high number of bright spots). This also causes their AM+PI outputs to be muddled with two many patterns. This adds a lot of difficulty in understandability of these attributes.

**How to choose the number of attributes** Assuming a suitable architecture for decoder $d$, simply tracking $\mathcal{L}_{if}, \mathcal{L}_{of}$ on training data can help rule out very small values of $J$ as they result in poorly trained decoder and relatively poor fidelity of $g$. One can also qualitatively analyze the generated explanations from the training data to tune $J$ to a certain extent. Too small values of $J$ can result in attributes encoding patterns for too many classes, which affects negatively their understandability. It is more tricky and subjective to tune $J$ once it becomes large enough so that $\mathcal{L}_{if}, \mathcal{L}_{of}$ are optimized well. The upper threshold of choosing $J$ is subjective and highly affected by how many attributes the user can keep a tab on or what fidelity user considers reasonable enough. It is possible that due to enforcement of conciseness, even for high value of $J$, only a small subset of attributes are relevant for interpretations. Nevertheless, for high $J$ value, there is a risk of ending up with too many attributes or class-attribute pairs to analyze.

It is important to notice that it is possible to select $J$ from the training set only by suing a cross-validation strategy. In practise, it seems reasonable to agree on smallest value of $J$ for which the increase of the cross-validation fidelity estimate drops dramatically, since further increase of $J$ would generate less understandable attributes with very little gain in fidelity.

| | $\mathcal{L}_{if}$ (train) | $\mathcal{L}_{of}$ (train) | Fidelity (test) (%) |
|---|---|---|---|
| $J = 4$ | 0.058 | 0.57 | 87.4 |
| $J = 8$ | 0.053 | 0.23 | 97.5 |
| $J = 25$ | 0.029 | 0.16 | 98.8 |

Table 4: Effect of $J$ on losses and fidelity for MNIST with LeNet.

| | $\mathcal{L}_{if}$ (train) | $\mathcal{L}_{of}$ (train) | Fidelity (test) (%) |
|---|---|---|---|
| $J = 4$ | 0.094 | 2.08 | 19.5 |
| $J = 8$ | 0.079 | 1.48 | 57.6 |
| $J = 24$ | 0.069 | 0.34 | 90.8 |

Table 5: Effect of $J$ on losses and fidelity for QuickDraw with ResNet.

## Disagreement analysis

In this part, we analyse in detail the "disagreement" between the predictor $f$ and the interpreter $g$.

First, it is important to notice that achieved by the proposed FLINT architecture fidelity of $98.9\%$ on MNIST and $90.8\%$ on QuickDraw is very reasonable compared to typical values reported in the literature. The disagreement between $f$ and $g$ is then close to $1.1\%$ and $9.2\%$, respectively. To the best of our knowledge, the highest reported fidelity for an explanation method on MNIST is 96.7% in the work of (Bang et al. 2019), which is 3 times more disagreement than FLINT. Recent methods for global interpretations (e.g. Lakkaraju et al. 2019) or robust local interpretations (e.g. Lakkaraju, Arsov, and Bastani 2020) report fidelity around $80\%$ for tabular datasets. The above systems and disagreement rates, as well as further interpretability frameworks are obviously not comparable with FLINT since they crucially differ either while reporting results on different data or from a systemic viewpoint on one of the following axes: (i) treating the model as a black-box, (ii) scope of interpretation (i.e., local vs. global), (iii) means of interpretation (human-understandable features), (iv) having interpretability as a part of learning objective. Nevertheless, our results emphasize that FLINT disagreement numbers are reasonably low.

Second, it is important to analyze in detail the type and the reason of disagreement between $f$ and $g$, especially for QuickDraw data. Thus, for the 'Apple' class, the only three disagreement samples for which $f$ delivers correct prediction (plotted in Fig. 13) are not resembling apples at all. We propose an original analysis approach that consists in calculating a *robust centrality measure*—the projection depth—of these three samples as well as of another $100$ training samples w.r.t. the $8000$ training 'Apple' samples, plotted in Fig. 14. To that purpose, we use the notion of projection depth (Zuo and Serfling 2000) (see also (Mosler 2013)) for a sample $\boldsymbol{x} \in \mathbb{R}^d$ w.r.t. a dataset $\boldsymbol{X}$ which is defined as follows:

$$D(\boldsymbol{x}|\boldsymbol{X}) = \left(1 + \sup_{\boldsymbol{p}\in\mathcal{S}^{d-1}} \frac{|\langle \boldsymbol{p}, \boldsymbol{x}\rangle - \text{med}(\langle \boldsymbol{p}, \boldsymbol{X}\rangle)|}{\text{MAD}(\langle \boldsymbol{p}, \boldsymbol{X}\rangle)}\right)^{-1}, \quad (1)$$

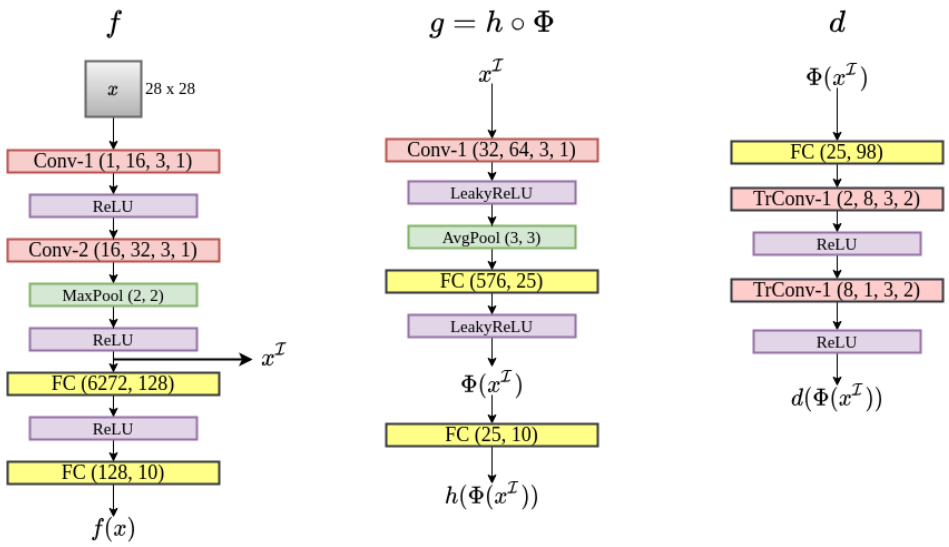



Figure 9: Architecture of networks based on LeNet (LeCun et al. 2015). Conv (a, b, c, d), TrConv (a, b, c, d) denote a convolutional, transposed convolutional layer respectively with number of input maps a, number of output maps b, kernel size c $\times$ c and stride size d. FC(a, b) denotes a fully-connected layer with number of input neurons a and output neurons b. MaxPool(a, a) denotes window size a $\times$ a for the max operation. AvgPool(a, a) denotes the output shape a $\times$ a for each map

with $\langle \cdot, \cdot \rangle$ denoting scalar product (and thus $\langle \boldsymbol{p}, \boldsymbol{X} \rangle$ being a vector of projection of $\boldsymbol{X}$ on $\boldsymbol{p}$) and med and MAD being the univariate median and the median absolute deviation form the median. Fig. 14 confirms the visual impression that these 3 disagreement samples are outliers (since their depth in the training class is low).

For a given sample with disagreement, if the class predicted by $f$ is among the top predicted classes of $g$, the disagreement is acceptable to some extent as the attributes can still potentially interpret the prediction of $f$. The worse kind of samples for disagreement are the ones where predicted by $f$ class is not among the top $g$ predicted classes, and even worse are where, in addition to this, $f$ predicts the true label. We thus compute the top-$k$ accuracy (for $k = 2, 3, 4$) on QuickDraw with ResNet, which for the default parameters described in the main paper, achieves a top-2 accuracy of $94.7\%$, top-3 accuracy $96.9\%$, and top-4 accuracy $98.2\%$. E.g., there are only $141$ (i.e. only $0.7\%$) of cases where the correctly predicted by $f$ class is not in top-3 predicted by $g$ classes. Fig. 15 depicts 26 such cases for 'Cat' class to illustrate their logical dissimilarity. Being a complex model, the ResNet-based predictor $f$ still manages to learn to distinguish these cases (while $g$ does not), but in a way $g$ does not manage at all to explain. Eventually, exploiting disagreement of $f$ and $g$ could be used as a means to measure trustworthiness. Deepening this issue is left for future works.

## Additional Results for Global Interpretations

We depict the interpretation of other attributes on QuickDraw & MNIST in Fig. 16 and Fig. 17 respectively. For each class-attribute pair, following the main paper, we select 3 maximum activating samples, analyze it using AM+PI output. One can observe the consistency of patterns detected by attributes across samples in a single pair, and even across different pairs (for the same attribute). It is also notable that there are some attributes whose patterns are harder to understand ($\phi_1, \phi_2$ in figure 16) however they rarely occur when training with autoencoder.

**Using decoder and input attribution** Although we consider AM+PI as the primary tool for analyzing patterns encoded by attributes (for MAS of each class-attribute), other tools like input attribution and decoder $d$ can also be helpful in deeper understanding of the attributes. We illustrate the use of these tools for certain example class-attribute pairs on QuickDraw in Fig. 18 and 19. Note that as discussed in the main paper, these tools are not guaranteed to be always insightful, but their use can help in some cases.

Fig. 18 depicts example class-attribute pairs where decoder $d$ contributes in understanding of attributes. The with $\phi_j$ column denotes the reconstructed sample $d(\Phi(x^{\mathcal{I}}))$ for the maximum activating sample $x$ under consideration. The without $\phi_j$ column is the reconstructed sample $d(\Phi(x^{\mathcal{I}})\backslash j)$ with the effect of attribute $\phi_j$ removed for the sample under consideration ($\phi_j(x^{\mathcal{I}}) = 0$). For eg. $\phi_1, \phi_{23}$, strongly relevant for Cat class, detect similar patterns, primarily related to the face and ears of a cat. The decoder images suggest that $\phi_1$ very likely is more responsible for detecting the left ear of cat and $\phi_{23}$, the right ear. Similarly analyzing decoder images for $\phi_{22}$ in the third row reveals that it is likely has a preference for detecting heads present towards the right side of the image. This is certainly not the primary pattern it detects as it mainly detects blotted textures, but it certainly carries information about head location to the decoder.

Fig. 19 depicts example class-attribute pairs where input attribution contributes in understanding of attributes. We use Guided Backpropagation (Springenberg et al. 2014) (GBP)

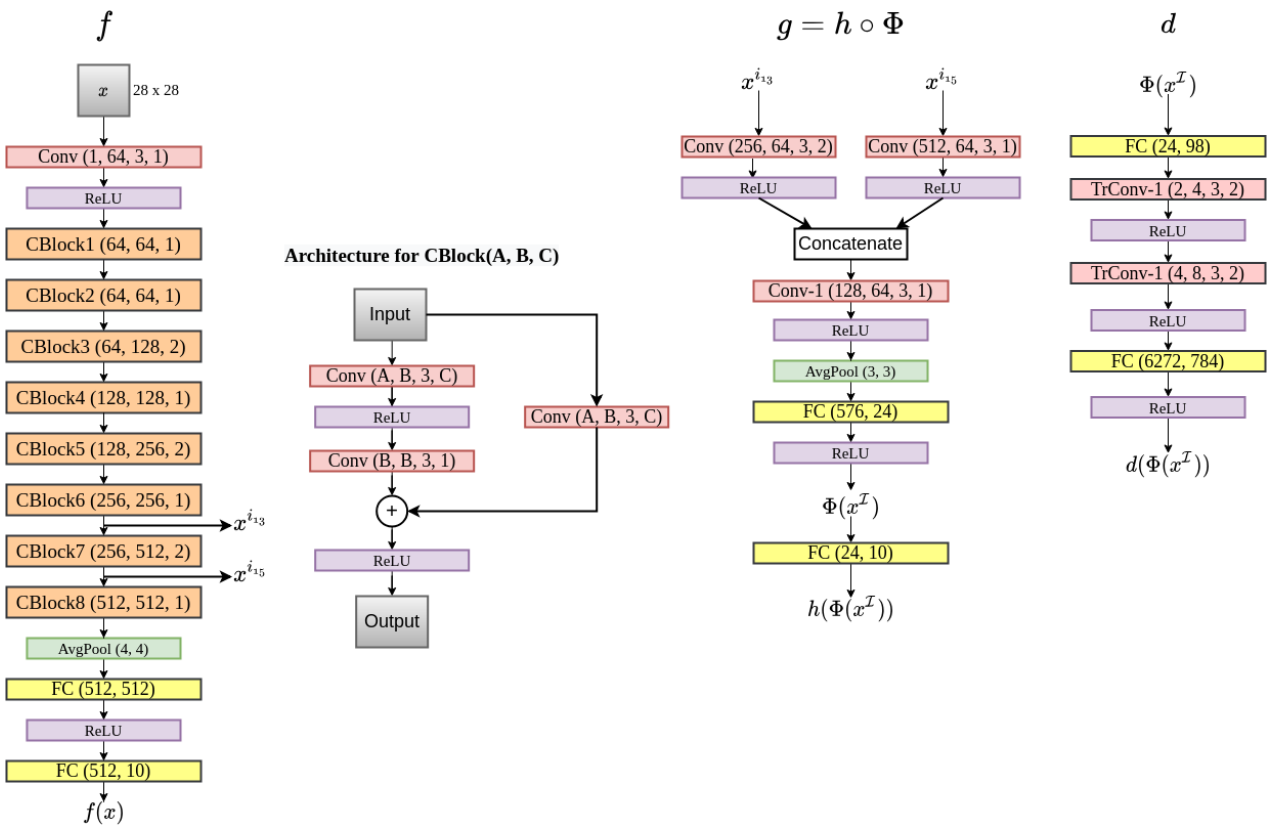



Figure 10: Architecture of networks for experiments on QuickDraw with network based on ResNet (He et al. 2016). Conv (a, b, c, d), TrConv (a, b, c, d) denote a convolutional, transposed convolutional layer respectively with number of input maps a, number of output maps b, kernel size c $\times$ c and stride size d. FC(a, b) denotes a fully-connected layer with number of input neurons a and output neurons b. AvgPool(a, a) denotes the output shape a $\times$ a for each map. Notation for CBlock is explained in the figure.

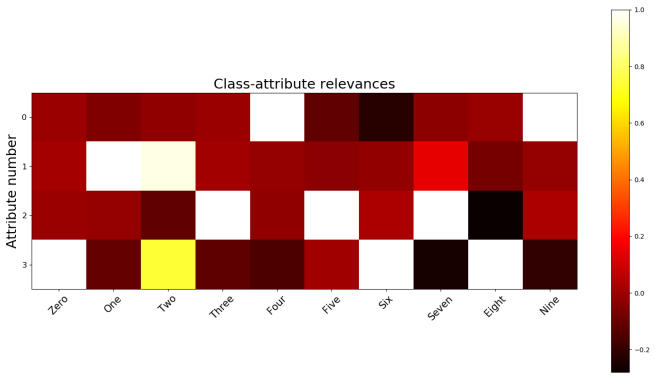



Figure 11: Global class attribute relevances for model with $J = 4$ on MNIST.

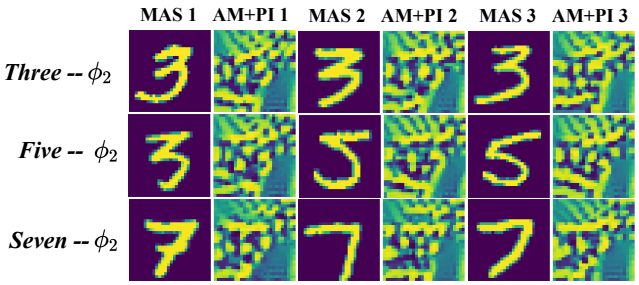



Figure 12: Interpretation for attribute $\phi_2$ for model learn on MNIST with $J = 4$.

as input attribution method for ResNet on QuickDraw. It mainly assists in adding more support to our previously developed understanding of attributes. For example, analyzing $\phi_5$ (relevant for Dog, Lion) based on AM+PI outputs suggested that it mainly detects curves similar to dog ears. The GBP output support this understanding as the most salient regions of the map correspond to curves similar to dog ears.

Fig. 20 shows the global class-attribute relevances (for all pairs) for QuickDraw & MNIST. Fig 21 shows relevances of attributes for 80 random test samples, on QuickDraw and MNIST. One can notice the similarities in the aggregated global relevances in fig. 20 and relevances for test samples in fig. 21. This consistency in action of attributes for test samples further supports our understanding from the generated interpretations.

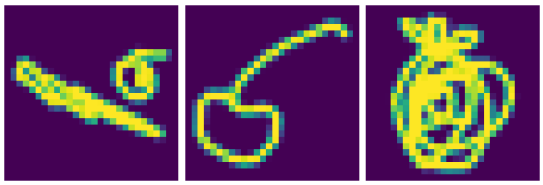


Figure 13: The three 'Apple' class samples classified correctly by $f$ but not by $g$.

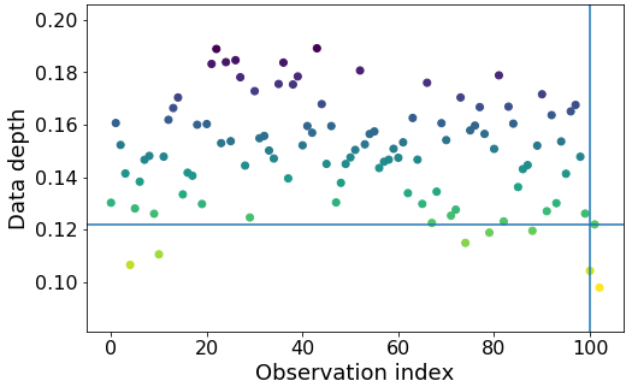



Figure 14: Projection data depth calculated with (1) w.r.t. the 8000 'Apple' training sample for 100 'Apple' test samples and for the three (observation indices 101–103) 'Apple' class samples classified correctly by $f$ but not by $g$.

**Global interpretation for Non-AE model** Fig. 22 shows 4 attributes and the corresponding class-attribute pairs for model trained without autoencoder on QuickDraw with ResNet, that is $\gamma = 0$. As remarked in the main paper, the patterns encoded attributes learnt for this model do not show much consistency for different samples. This is the case for most attributes in the figure (class-attribute pairs for $\phi_{10}, \phi_{12}, \phi_{22}$, 3rd row onwards).

## Additional results on FashionMNIST

We compute additional results on FashionMNIST (Xiao, Rasul, and Vollgraf 2017). We use the same architecture for the networks and hyperparameters as for MNIST (based on LeNet). The accuracies and fidelity values are reported in Tabs. 6 and 7. Global interpretations of sample attributes is depicted in Fig. 23.

| Dataset | Architecture | System | Accuracy (%)↑ |
|---|---|---|---|
| FashionMNIST | LeNet | BASE-$f$ | 90.3 |
| | | FLINT-$f$ | 90.6 |
| | | FLINT-$g$ | 87.5 |

Table 6: Accuracies on FashionMNIST.

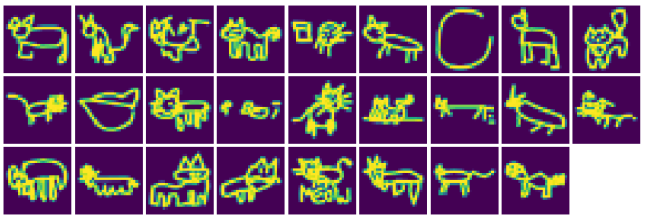


Figure 15: 26 samples from 'Cat' class which are not in top3 $f$-predicted classes.

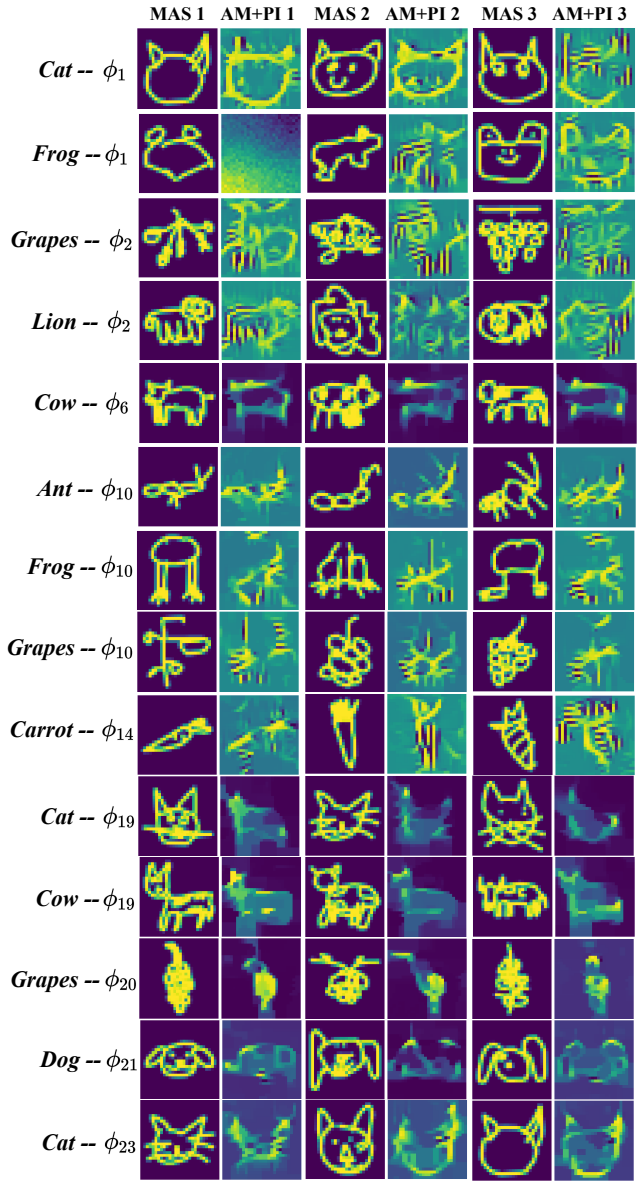



Figure 16: Global overview of learnt attributes on QuickDraw with ResNet (does not contain attributes in main paper). Each row is composed of analysis of single class-attribute pair with 3 MAS and their AM+PI outputs.

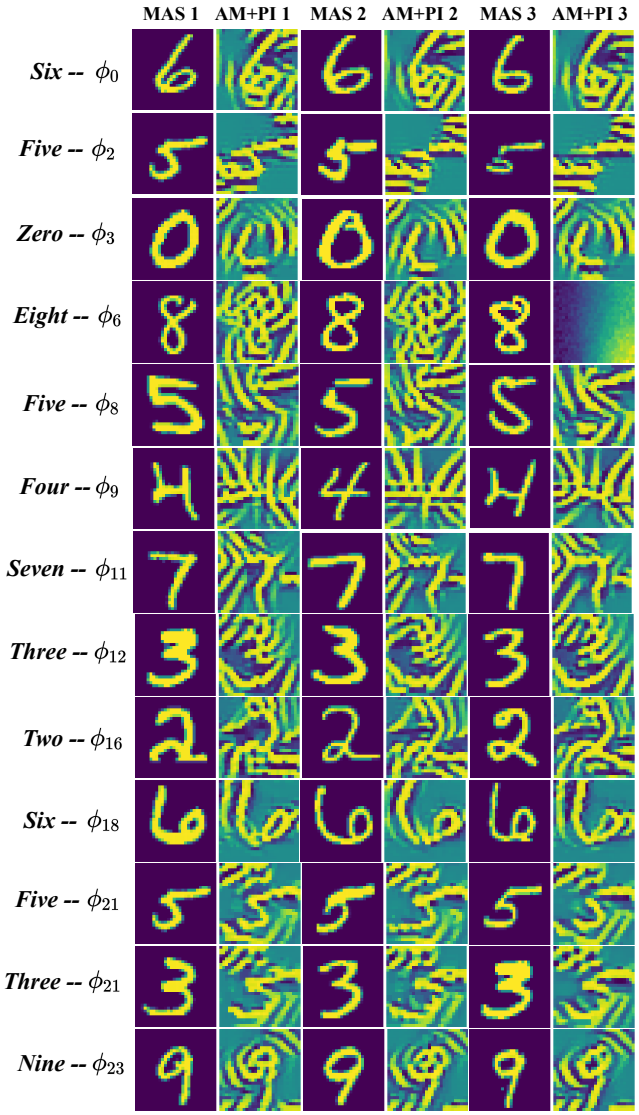



Figure 17: Global overview of attributes learnt on MNIST with LeNet. Each row is composed of analysis of single class-attribute pair with 3 MAS and their AM+PI outputs.

| Dataset | Architecture | Fidelity (%) |
|---|---|---|
| FashionMNIST | LeNet | 91.5 |

Table 7: Fidelity result for FashionMNIST test set

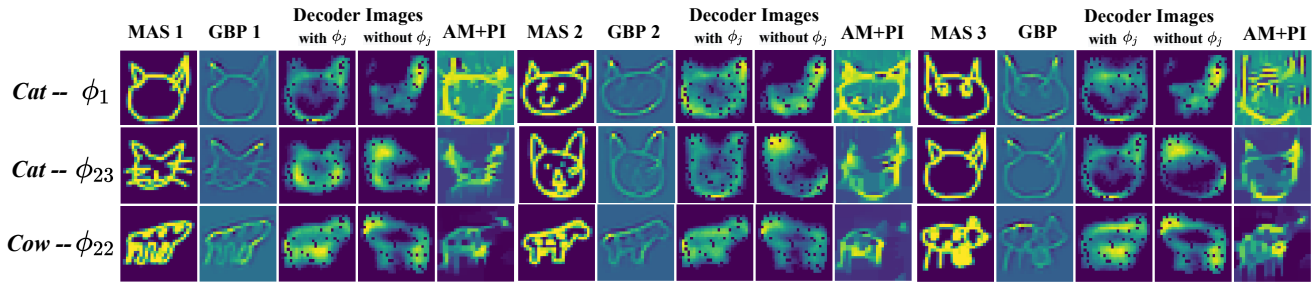



Figure 18: Examples of class-attribute pairs on QuickDraw, where decoder assists in understanding of patterns for the attribute.

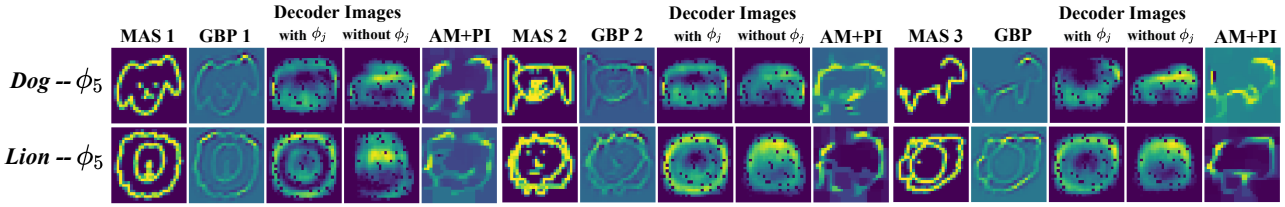
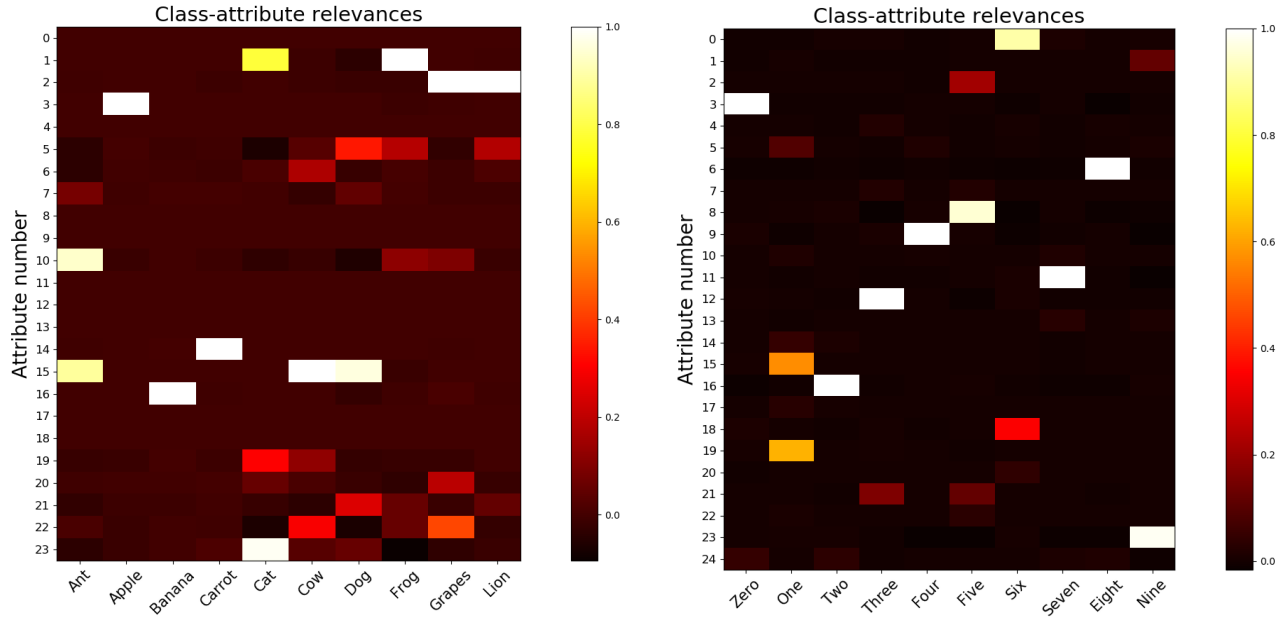


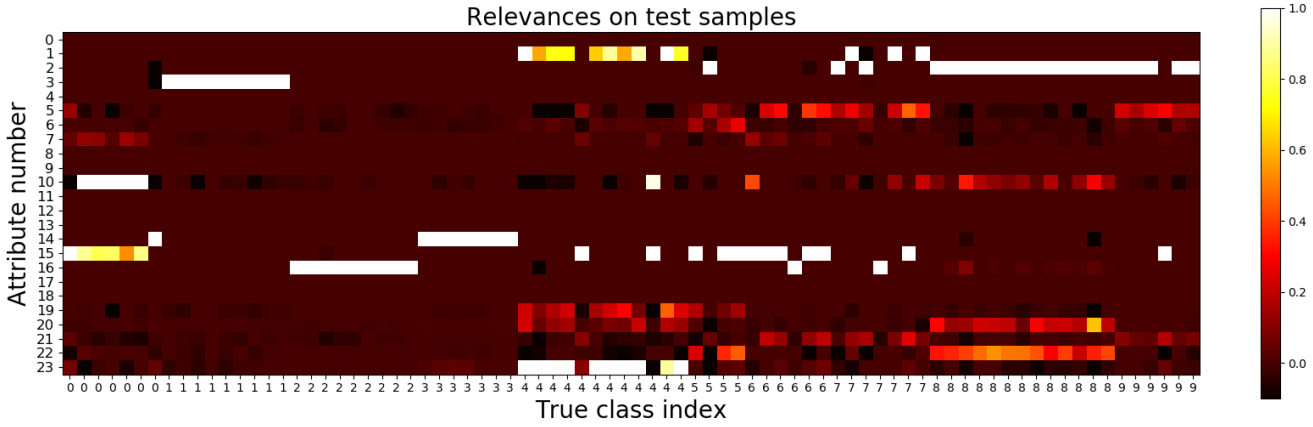

Figure 19: Examples of class-attribute pairs on QuickDraw, where input attribution (GBP) assists in understanding of encoded patterns for the attribute. GBP stands for Guided Backpropagation.



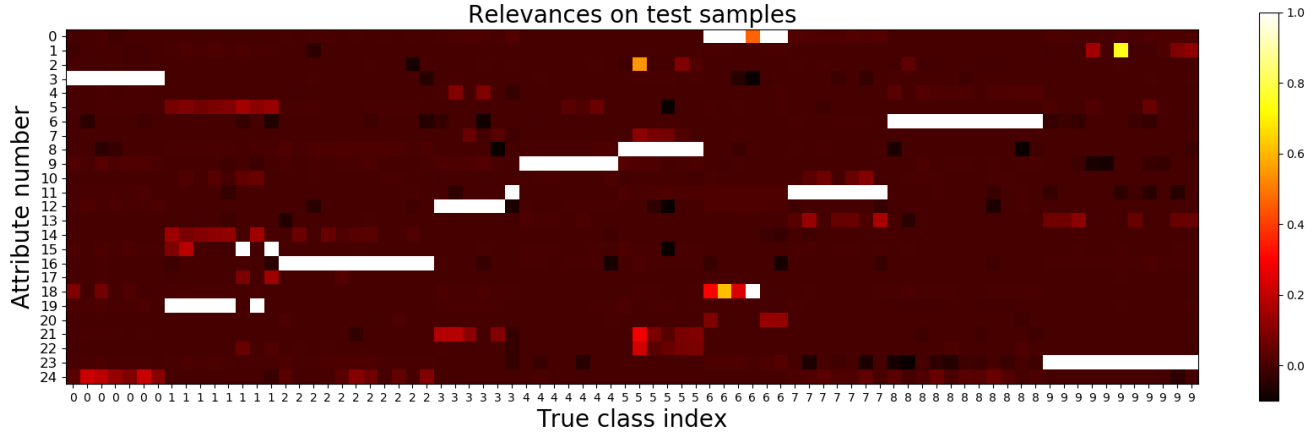

(a) Relevances for all class-attribute pairs on QuickDraw

(b) Relevances for all class-attribute pairs on MNIST

Figure 20: Global class-attribute relevances for QuickDraw & MNIST. On x-axis is the class labels and on y-axis is the attribute numbers.




(a) Relevances for 80 random test samples on QuickDraw shuffled according to classes. x-axis denotes the true class label for the sample. 0: Ant, 1: Apple, 2: Banana, 3: Carrot, 4: Cat, 5: Cow, 6: Dog, 7: Frog, 8: Grapes, 9: Lion




(b) Relevances for 80 random test samples on MNIST shuffled according to classes. x-axis denotes the true class label for the sample.

Figure 21: Local relevances of the attributes for a selection of test samples.

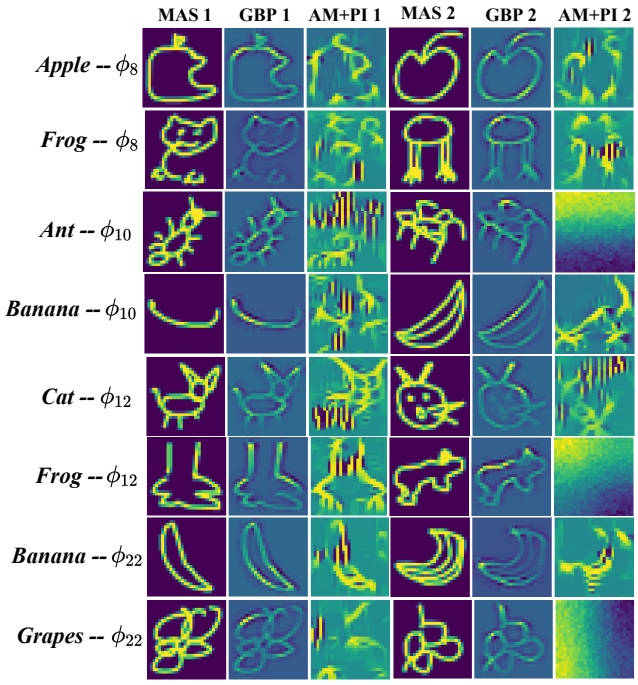



Figure 22: Global overview of sample attributes learnt on QuickDraw with ResNet but without any autoencoder ($\gamma = 0$). GBP stands for Guided Backpropagation.

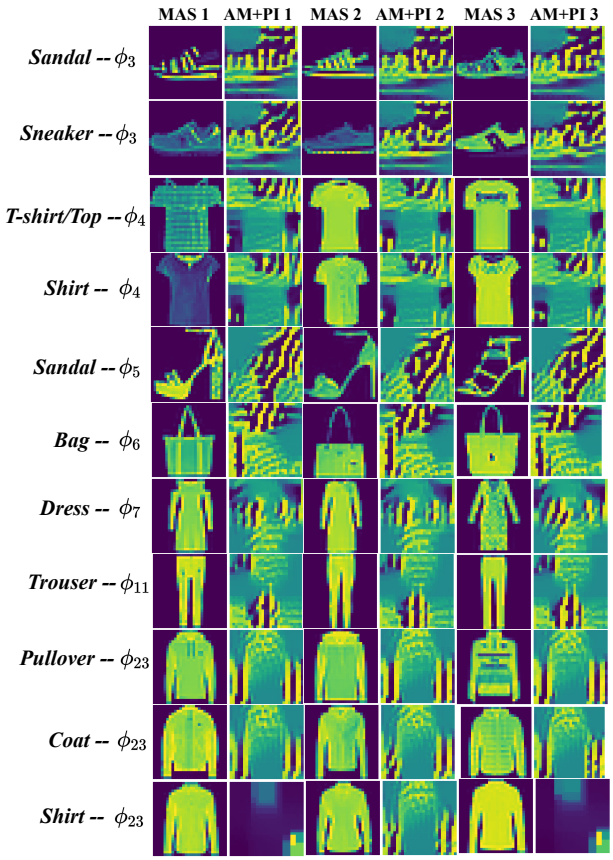



Figure 23: Global overview of sample attributes learnt on FashionMNIST with LeNet. Each row is composed of analysis of single class-attribute pair with 3 MAS and their AM+PI outputs