Mining *Data Impressions* from Deep Models as Substitute for the Unavailable Training Data

Gaurav Kumar Nayak, *Graduate Student Member, IEEE,* Konda Reddy Mopuri, Saksham Jain, and Anirban Chakraborty, *Member, IEEE*

Abstract—Pretrained deep models hold their learnt knowledge in the form of model parameters. These parameters act as "memory" for the trained models and help them generalize well on unseen data. However, in absence of training data, the utility of a trained model is merely limited to either inference or better initialization towards a target task. In this paper, we go further and extract synthetic data by leveraging the learnt model parameters. We dub them *Data Impressions*, which act as proxy to the training data and can be used to realize a variety of tasks. These are useful in scenarios where only the pretrained models are available and the training data is not shared (e.g., due to privacy or sensitivity concerns). We show the applicability of data impressions in solving several computer vision tasks such as unsupervised domain adaptation, continual learning as well as knowledge distillation. We also study the adversarial robustness of lightweight models trained via knowledge distillation using these data impressions. Further, we demonstrate the efficacy of data impressions in generating data-free Universal Adversarial Perturbations (UAPs) with better fooling rates. Extensive experiments performed on benchmark datasets demonstrate competitive performance achieved using data impressions in absence of original training data.

Index Terms—Data Impressions, Proxy Data, Synthetic Transfer Set, Surrogate Data, Absence of Training Data, Knowledge Distillation, Universal Adversarial Perturbations, Continual Learning, Unsupervised Domain Adaptation

1 Introduction

Supervised learning typically requires large volumes of labelled data. Training of sophisticated deep neural networks (DNNs) often involves learning from thousands (MNIST [1], CIFAR [2]) (sometimes millions, e.g. ImageNet [3]) of data samples. Despite their ability to train complex models, these training datasets pose practical challenges. These datasets (i) are often huge in size (e.g. ImageNet [3]), (ii) are proprietary, and (iii) involve privacy concerns (e.g. biometric, healthcare data). Hence, in practice, public access to the data samples used for training may not always be feasible. Instead, the resulting trained models can be made available relatively easily. For instance, Facebook's Deepface [4] model is trained over 4M confidential face images.

However, in the absence of training data, a trained model has limited utility for adapting it to a related task. In this scenario, the best thing one can do is utilizing the trained layers as a better initialization for a fresh training. In other words, unavailability of the training data restricts the transfer learning possibilities to a mere pretraining. Because of this, applications with more practical significance such as Knowledge Distillation (KD) and Unsupervised Domain Adaptation can not be realised in the absence of the training data. For instance, in the KD framework, to compress a sophisticated (Teacher) Neural Network into a light weight (Student) one, training data is required as the transfer set. Matching the input output behaviour of the models (despite their architectural differences) which is the key for model compression can not take place in the absence of training data. Given no prior information about the underlying training data, it is challenging to compose a suitable transfer set to replace it. Similarly, for Unsupervised Domain Adaptation, data on which the source model is trained plays a vital role for an effective knowledge transfer. In most target scenarios, nontrivial performances can be achieved by suitably adapting the source models. All these possibilities get abolished when we do not have the training data along with the trained model. This leads to a massive under utilization of the training efforts. Therefore, in this paper we investigate for approaches that can craft proxy data for extending the utility of a trained model beyond pretraining. Specifically, we propose to utilize the given trained model itself for extracting the proxy data.

We consider the Convolutional Neural networks (CNNs) trained for object recognition. Starting from Knowledge Distillation, we explore data-free adaptation of the trained model in various application scenarios. Inspired from Mopuri *et al.* [5], we extract impressions of training data from the parameters of the trained CNN model. Note that with no external prior about the training data, we perform the proxy data synthesis required for the adaptation. We extract the inter-class similarities from the CNN parameters and model the output (softmax) space of the classifier using a family of Dirichlet distributions. We sample these distributions and iteratively reconstruct the corresponding data samples in the input space from random initializations. Our approach extracts the proxy data samples from a trained model one

- *G. K. Nayak and A. Chakraborty are with the Department of Computational and Data Sciences, Indian Institute of Science, Bangalore, India.*
- *S. Jain is currently with the Department of Electrical and Computer Engineering, Duke University USA. He was affiliated with the Department of Computational and Data Sciences, Indian Institute of Science when this work was carried out.*
- *K. R. Mopuri is with the Department of Computer Science and Engineering, Indian Institute of Technology Tirupati.*

For all correspondence: Anirban Chakraborty (anirban@iisc.ac.in)

at a time. The recovered synthetic data samples are named *Data Impressions* (DIs) as they are the impressions of actual data extracted from the model. Note that the extraction process neither requires original training data nor any prior information, and the extracted samples act as a proxy in the absence of original data. Importantly, the extraction of the impressions is agnostic to the downstream application. In other words, the same method can be applied directly across multiple applications. This observation denotes that they capture generic knowledge about the training dataset suitable for adapting to various application.

One way to ensure the effectiveness of the extracted surrogate samples is via generalization. That is, by demonstrating that the extracted samples can be reliably used for adapting the model and generalize well on to the actual test data. Hence, for each adaptation we empirically verify the performance of the adapted models on the actual test datasets. In order to show the effectiveness of such generated data impressions, we leverage several computer vision applications that have faced problems arising from data-free set up. These problems have been tackled independently in the literature and various methods have already been proposed. We simply leverage these problems and propose solution strategies utilizing the aforementioned data impressions. We observe strong performances across all these tasks, thereby proving the utility of our data impressions as surrogates to the original training data.

Here we would like to emphasize that these applications are to demonstrate the effectiveness of the data impressions and prove that they are reliable surrogates for the original training data samples. Hence it may be unfair to compare the performance with the corresponding dedicated data-free solutions for the individual applications. Also, given the generic nature of the data impressions, they may be utilized in several other tasks apart from the ones that we discuss in this work.

The overall contributions of our work can be listed as follows:

- We propose the first and generic framework for data-free adaptation of trained neural networks via extracting proxy data samples, called 'Data Impressions'. We achieve this with no additional prior about the training data and without requiring any meta-data about the resulting feature distribution.
- We study the the extensive applicability of Data Impressions towards multiple applications such as Knowledge Distillation, Unsupervised Domain Adaptation, crafting Adversarial Perturbations, and Incremental Learning. We show that in the absence of original training data, Data Impressions can successfully train models that generalize well onto the actual test data.
- Further, we study the robustness properties of the student models trained on the Data Impressions against adversarial perturbations. Experimental results demonstrate that Data Impressions consistently uphold the robustness properties of the corresponding teachers.

Note that the framework for extracting the Data Impressions and their application Zero-Shot Knowledge Distillation were originally introduced in our earlier conference paper [6]. All the other contributions are novel additions to this extended article.

The rest of this paper is organised as follows: section 2 discusses the existing works that are related to this research, section 3 presents our approach for extracting the Data Impressions from a trained CNN classifier, section 4 demonstrates the effectiveness of the approach via learning multiple related tasks, section 5 discusses the major findings across experiments on different applications and finally section 6 summarizes the paper with conclusions.

2 Related Work

Our work introduces a novel problem of restoring training data from a trained deep model. It is broadly related to visualization works such as [7], [8]. However, the general objective driving visualization works is to identify the patterns in the stimuli for which the neurons maximally respond and thereby alleviate their black-box nature. Based on the gradient driven visualization ideas, Mopuri *et al.* [5] craft class representative samples, known as Class Impressions, from a trained CNN based classifier. Their objective is specific, which is to use these impressions for crafting adversarial perturbations in a data-free scenario. We extend this idea and make it a generic problem of extracting the samples that can substitute the training data. Further, we demonstrate the effectiveness of our Data Impressions by successfully learning diverse set of related tasks over the restored data. Specifically we perform Knowledge Distillation, UAP (Universal Adversarial Perturbation) generation, Domain Adaptation, and Incremental Learning. For ease of reference, we briefly introduce these tasks and compare our idea of using Data Impressions with the corresponding existing works.

Knowledge distillation: is a process of emulating a large model called *Teacher* by a lightweight model called *Student*. The teacher model generally has high complexity and is not preferred for real-time embedded platforms due to its large memory and computational requirements. In practice, networks which are compact and lightweight are preferred. Existing works use training data (e.g. [9], [10]) or meta data (e.g. [11]) extracted from the teacher for performing distillation. However, the proposed method transfers the knowledge without using either of them. To the best of our knowledge, our work (Nayak *et al.* [6]) is the first to demonstrate knowledge distillation in case where no training data is available. Contemporary to our work, Chen *et al.* [12] Micaelli *et al.* [13] and Addepalli *et al.* [14] also attempt to perform knowledge transfer in the data-free scenario. However, unlike our activation maximization approach, they train GAN-inspired generative models to learn the proxy or fake data samples required for the transfer. These methods train the GAN with multiple objectives to ensure learning (i) difficult pseudo (or proxy) samples on which the Teacher and Student differ ([13]), (ii) uniform distributions over the underlying classes ([14]), and (iii) samples predicted with a strong confidence by the Teacher model, ([12]) etc. so that the transfer performance is maximized. Note that [14] uses arbitrary but natural proxy data for transferring the knowledge. Another generative model known as KegNet [15] by

Yoo *et al.* also employs a conditional GAN framework along with a decoder objective for encouraging diversity in the fake images used for knowledge transfer. Unlike these GAN based approaches, our method do not involve such complex training procedures and do not require any "proxy" data samples as used in [14], thereby strictly adhering to the "zero-shot" paradigm.

On the other hand, recent works by Yin *et al.* [16] and Haroush *et al.* [17] attempt to synthesize the class conditional samples from a trained neural network model and enforce intuitive priors to improve the quality of the generated samples. Specifically they utilize the Batch Normalization (BN) layers' statistics such as feature mean and covariances extracted from the Teacher as the useful prior while synthesizing via maximizing the logit activations. Further, [16] also imposes natural image priors such as smoothness while synthesizing the pseudo samples. Similarly Shoukai *et al.* [18] present a conditional GAN framework for quantizing a trained Teacher model in the data-free scenario by learning fake images. Their method along with utilizing the Batch Norm statistics for matching the training data distribution, also uses the Knowledge Distillation and CrossEntropy objectives for compressing the Teacher model. That way these works can be thought of improvements to our method but restricted to invert the models that have BN layers. These methods by design, are restricted to specific network architectures that use the batchnorm layers and hence cannot be utilized for older architectures or recent models that do not include such layers. Our framework, on the other hand, are completely independent of the pretrained network architecture and hence are more widely applicable. Also, additionally in this work we perform robustness study on student models trained in the data-free setup.

Incremental Learning: Here the goal is to train a model using the samples of new classes without forgetting the knowledge gained from the old classes. With the limited memory constraints, several rehearsal based methods such as Rebuffi *et al.* [19] and Castro *et al.* [20], carefully store few samples from the old classes to avoid catastrophic forgetting. Pseudo-rehearsal methods like Shin *et al.* [21] avoid storing samples from old classes, but instead they learn a generator which is trained using old class data. Thus, there is an implicit assumption of the availability of the trained generator which is as good as having access to old class data. Moreover training of a generator has its own difficulty like mode collapse which requires proper handling.

In cases where samples belonging to old classes are unavailable and only have access to the pretrained model which is trained on those classes, above discussed methods perform poorly. LwF [22] by Li *et al.* tries to overcome this problem by only utilizing samples belonging to the new classes. The model is trained with these samples where cross entropy loss is used on new classes while distillation is applied on old classes. Recently, DMC [23] by Zhang *et al.* has shown great results by using auxiliary data and dual distillation loss. Their performance is dependent on how close the selected auxiliary data is to the training data distribution. Often domain knowledge is required for a careful selection of such data, which becomes a non-trivial task when no prior on the original training data is available except for a model trained with old classes. We, on the other hand, do not use any auxiliary data but instead generate Data Impressions from the model trained on old classes and use them with new class data to train the combined model.

Unsupervised Domain Adaptation: The goal in this task is to adapt the model trained on source data to predict the labels for the unlabelled target data. Most of the existing works such as [24], [25], [26], [27], [28] depend on the availability of both source and target data to perform the adaptation. However, recent work by Kundu *et al.* [29] overcomes this limitation, but only in the deployment stage. In the procurement stage, they require the source model to be trained not only on source training samples but also on negative source samples simulated with the help of source data. As a main difference to [29], we restrict ourselves to closed set domain adaptation and we leverage Data Impressions in the absence of source data to perform source-free domain adaptation. Please note that under this setup, our method is generic and can be applied on any trained source model.

Recently, Liang *et al.* [30] proposed a new method (SHOT), which aligns the target features to the source hypothesis in the absence of the source data and target labels. The source-trained model is composed of feature extractor and classifier. The classifier module (hypothesis) is frozen and the feature encoding module is finetuned to fit the hypothesis using information maximization loss along with pseudo labeling via self supervision. Even though they obtain promising results, their method is heavily dependent on the architecture of the source network. They require weight normalization in the fully connected layer of the classifier and batch normalization layer at the end of the feature extractor module. The performance on the target data drops significantly when such dependencies are not met. On the other hand, our method does not have any such architectural dependencies.

Another recent work by Kurmi *et al.* [31] proposed an end to end framework where generation of pseudo samples and their use for adaptation in the absence of source data are performed simultaneously. In the generation module of [31], samples are synthesized using conditional GAN by modelling the joint distribution of images and corresponding labels of the source data. Additionally the authors use an adversarial discriminator to close the gap between the distribution of generated samples to that of target samples. So, they require target data in order to generate pseudo-samples. Also, their overall loss function is a combination of multiple losses which requires careful balancing. Unlike theirs, our generation method is independent of the target data. We generate data impressions using only the pretrained source model and is generic as its synthesis does not depend on the target data. Moreover, we do not perform any complicated GAN training. As our synthesis of samples is done using a single loss function, the optimization is easy to handle.

Universal Adversarial Perturbations (UAPs): UAPs or Image agnostic adversarial perturbations are structured, mild noises that upon adding to the data, can confuse deep classifiers and enforce them to predict incorrectly. The training data (e.g. Moosavi-Dezfooli [32]) is generally required to craft the UAP. Mopuri *et al.* [33], [34], for the first time, presented a data-free approach for crafting UAPs

using an activation maximization objective. Later they proposed Class Impressions [35] as a way to reduce the gap between data-free and data-driven approaches. Proposed Data Impressions capture the patterns from the training data better than their class impressions and thereby can craft UAPs with better fooling ability.

Summary of differences with data-free methods: Several methods such as [12], [13], [15], [17], [18], [23], [29], [30], [31], [35] have been proposed in the data-free set up towards different applications which are specifically designed. However, such methods are dedicated data-free solutions for individual applications. Hence, they are application specific where the data generation process is tied to the task at hand. On the other hand, our proposed data impressions are synthesized without considering any downstream target task. We evaluate their efficacy by exploring their applications on different downstream tasks. We demonstrate that such impressions are indeed true substitutes of original training data samples and are suitable to be utilized across different applications.

Recently, Yin *et al.* [16] also shows the utility of their pseudo samples on data-free pruning and continual learning besides their application in knowledge distillation. Their method *'Deep Inversion'* is an extension of *'Deep Dream'* [36] where they additionally regularize the feature distribution of generated data by matching the batchnorm statistics. Their method assumes the presence of batchnorm layers which are prevalent only in the modern networks. Hence, the performance of their method is heavily dependent on the number of batch norm layers in the intermediate layers of the trained classifier. They further boost their performance by an iterative method *'Adaptive DeepInversion'* that generates samples which cause teacher-student disagreement. As the student is involved in the loop, this scheme is application dependent and is also very similar to [13]. Their overall loss optimization contains a sum of many regularization losses, where finding appropriate weightage of the individual losses is troublesome. On the other hand, our generation strategy of data impressions does not depend on batchnorm layers in the trained classifier. This makes our framework independent of the pretrained network architecture and hence is more widely applicable. In other words, our method is not only application-independent but also architecture-independent. Apart from data-free knowledge distillation, we also show the utility of our generated impressions on a diverse set of applications which are disjoint in comparison to [16] (such as source-free unsupervised domain adaptation, and data-free universal adversarial perturbations). Moreover, for the first time we study the robustness properties of a student distilled in a data-free scenario.

We now discuss in detail our proposed approach for synthesizing data impressions.

3 Proposed Approach: Extracting Data Impressions From Trained Models

In this section we describe the proposed method to extract samples from a *Trained* neural network model, which can act as substitute to the original training data. We first model the output (softmax) space of the *Trained* classifier using a probability distribution. Then we sample softmax vectors from this distribution. For each softmax vector, we generate corresponding input via iteratively updating a random input. Modelling of the softmax space and estimation of the distribution parameters is explained in sec. 3.1 while the procedure to generate the samples from the sampled softmax vectors is described in sec. 3.2

3.1 Modelling the Data in Softmax Space

In this work, we deal with the scenario where we have no access to (i) any training data samples, or (ii) metadata extracted from it (e.g. [11]). In order to tackle this, our approach taps the memory (learned parameters) of the *Trained* model and synthesizes pseudo samples to represent the underlying data distribution on which it is trained. Since these are the impressions of the training data extracted from the trained model, we name these synthesized input representations as Data Impressions. We argue that these can serve as effective surrogates for the training samples, which can be used to perform several tasks such as knowledge distillation, incremental learning, and unsupervised domain adaptation.

In order to craft the Data Impressions, we model the output (softmax) space of the *Trained* model. Let $\boldsymbol{s} \sim p(\boldsymbol{s})$, be the random vector that represents the neural softmax outputs of the *Trained* model, T with parameters θ_T. We model $p(\boldsymbol{s}^k)$ belonging to each class k, using a Dirichlet distribution which is a distribution over vectors whose components are in $[0, 1]$ range and their sum is 1. Thus, the distribution to represent the softmax outputs $\boldsymbol{s}^k$ of class k would be modelled as, $Dir(K, \boldsymbol{\alpha}^k)$, where $k \in \{1 \dots K\}$ is the class index, K is the dimension of the output probability vector (number of categories in the recognition problem), and $\boldsymbol{\alpha}^k$ is the concentration parameter of the distribution modelling class k. The concentration parameter $\boldsymbol{\alpha}^k$ is a K dimensional positive real vector, i.e, $\boldsymbol{\alpha}^k = [\alpha_1^k, \alpha_2^k, \dots, \alpha_K^k]$, and $\alpha_i^k > 0, \forall i \in \{1, 2, \dots K\}$.

Concentration Parameter ($\boldsymbol{\alpha}$): Since the sample space of the Dirichlet distribution is interpreted as a discrete probability distribution (over the labels), intuitively, the concentration parameter $(\boldsymbol{\alpha})$ can be thought of as determining how "concentrated" the probability mass of a sample from a Dirichlet distribution is likely to be. With a value much less than 1, the mass will be highly concentrated in only a few components, and all the rest will have almost zero mass. On the other hand, with a value much greater than 1, the mass will be dispersed almost equally among all the components.

Obtaining prior information for the concentration parameter is not straightforward. The parameter cannot be the same for all components since this results in all sets of probabilities being equally likely, which is not a realistic scenario. For instance, in case of CIFAR-10 dataset, it would not be meaningful to have a softmax output in which the dog class and plane class have the same confidence (since they are visually dissimilar). Also, same α_i values denote the lack of any prior information to favour one component of sampled softmax vector over the other. Hence, the concentration parameters should be assigned in order to reflect the similarities across the components in the softmax vector. Since these components denote the underlying categories

in the recognition problem, $\boldsymbol{\alpha}$ should reflect the visual similarities among them.

Thus, we resort to the *Trained* network for extracting this information. We compute a normalized class similarity matrix (C) using the weights W connecting the final (softmax) and the pre-final layers. The element $C(i,j)$ of this matrix denotes the visual similarity between the categories i and j in $[0,1]$. Thus, a row $\boldsymbol{c}_k$ of the class similarity matrix (C) gives the similarity of class k with each of the K categories (including itself). Each row $\boldsymbol{c}_k$ can be treated as the concentration parameter $(\boldsymbol{\alpha})$ of the Dirichlet distribution (Dir), which models the distribution of output probability vectors belonging to class k.

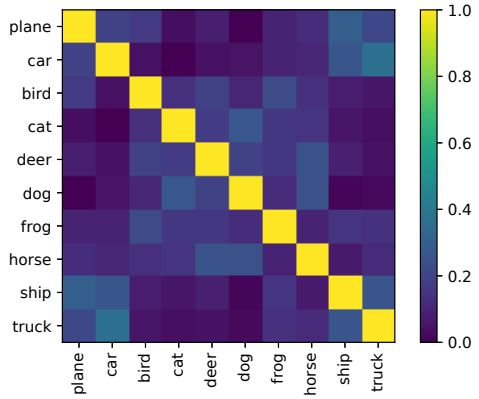

Fig. 1. Class similarity matrix computed for the Teacher model trained over CIFAR-10 dataset. Note that the class labels are mentioned and the learned similarities are meaningful.

Class Similarity Matrix: The class similarity matrix C is calculated as follows. The final layer of a typical recognition model will be a fully connected layer with a softmax non-linearity. Each neuron in this layer corresponds to a class (k) and its activation is treated as the probability predicted by the model for that class. The weights connecting the previous layer to this neuron $(\boldsymbol{w}_k)$ can be considered as the template of the class k learned by the *Trained* network. This is because the predicted class probability is proportional to the alignment of the pre-final layer's output with the template $(\boldsymbol{w}_k)$. The predicted probability peaks when the pre-final layer's output is a positive scaled version of this template $(\boldsymbol{w}_k)$. On the other hand, if the output of the pre-final layer is misaligned with the template $\boldsymbol{w}_k$, the confidence predicted for class k is reduced. Therefore, we treat the weights $\boldsymbol{w}_k$ as the class template for class k and compute the similarity between classes i and j as:

$$C(i,j) = \frac{\boldsymbol{w}_i^T \boldsymbol{w}_j}{\|\boldsymbol{w}_i\| \|\boldsymbol{w}_j\|} \tag{1}$$

Since the elements of the concentration parameter have to be positive real numbers, we further perform a min-max normalization over each row of the class similarity matrix. The visualization of the class similarity matrix calculated from a CIFAR-10 trained model is shown in Figure 1.

3.2 Crafting *Data Impressions* via Dirichlet Sampling

Once the parameters K and $\boldsymbol{\alpha}^k$ of the Dirichlet distribution are obtained for each class k, we can sample class probability (softmax) vectors, which respect the class similarities as learned by the *Trained* network. Using the optimization procedure in eq. (2) we obtain the input representations corresponding to these sampled output class probabilities. Let $Y^k = [\boldsymbol{y}_1^k, \boldsymbol{y}_2^k, \ldots, \boldsymbol{y}_N^k] \in \mathbb{R}^{K \times N}$, be the N softmax vectors corresponding to class k, sampled from $Dir(K, \boldsymbol{\alpha}^k)$ distribution. Corresponding to each sampled softmax vector $\boldsymbol{y}_i^k$, we can craft a Data Impression $\bar{x}_i{}^k$, for which the *Trained* network predicts a similar softmax output. We achieve this by optimizing the objective shown in eq. (2). We initialize $\bar{x}_i^k$ as a random noisy image and update it over multiple iterations till the cross-entropy loss between the sampled softmax vector $(\boldsymbol{y}_i^k)$ and the softmax output predicted by the *Trained* model T, is minimized.

$$\bar{x}_i{}^k = \underset{x}{\operatorname{argmin}}\ L_{CE}(\boldsymbol{y}_i^k, T(x, \theta_T, \tau)) \tag{2}$$

where τ is the temperature used in the softmax layer. The process is repeated for each of the N sampled softmax probability vectors in Y^k, $k \in \{1 \ldots K\}$.

Algorithm 1 Generation of Data Impressions

Input: Trained classifier T
N: number of DIs crafted per category,
$[\beta_1, \beta_2, ..., \beta_B]$: B scaling factors,
τ: Temperature
Output: $\bar{X}$: Data Impressions
Obtain K: number of categories from T
Compute the class similarity matrix $C = [\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_K^T]$ as in eq. (1)
$\bar{X} \leftarrow \emptyset$
for *k=1:K* **do**
  Set the concentration parameter $\boldsymbol{\alpha}^k = \mathbf{c}_k$
  for *b=1:B* **do**
    for *n=1:*$\lfloor N/B \rfloor$ **do**
      Sample $\boldsymbol{y}_n^k \sim Dir(K, \beta_b \times \boldsymbol{\alpha}^k)$
      Initialize $\bar{x}_n^k$ to random noise and craft $\bar{x}_n^k = \underset{x}{\operatorname{argmin}}\ L_{CE}(\boldsymbol{y}_n^k, T(x, \theta_T, \tau))$
      $\bar{X} \leftarrow \bar{X} \cup \bar{x}_n^k$
    end
  end
end

Scaling Factor (β): The probability density function of the Dirichlet distribution for K random variables is a $K-1$ dimensional probability simplex that exists on a K dimensional space. In addition to parameters K and $\boldsymbol{\alpha}$ as discussed in section 3.1, it is important to discuss the significance of the range of $\alpha_i \in \boldsymbol{\alpha}$, in controlling the density of the distribution. When $\alpha_i < 1,\ \forall i\ \epsilon[1,K]$, the density congregates at the edges of the simplex [37], [38]. As their values increase (when $\alpha_i > 1, \forall i \in [1,K]$), the density becomes more concentrated on the center of the simplex [37], [38]. Thus, we define a scaling factor $(\boldsymbol{\beta})$ which can control the range of the individual elements of the concentration parameter, which in turn decides regions in the simplex from which sampling is performed. This

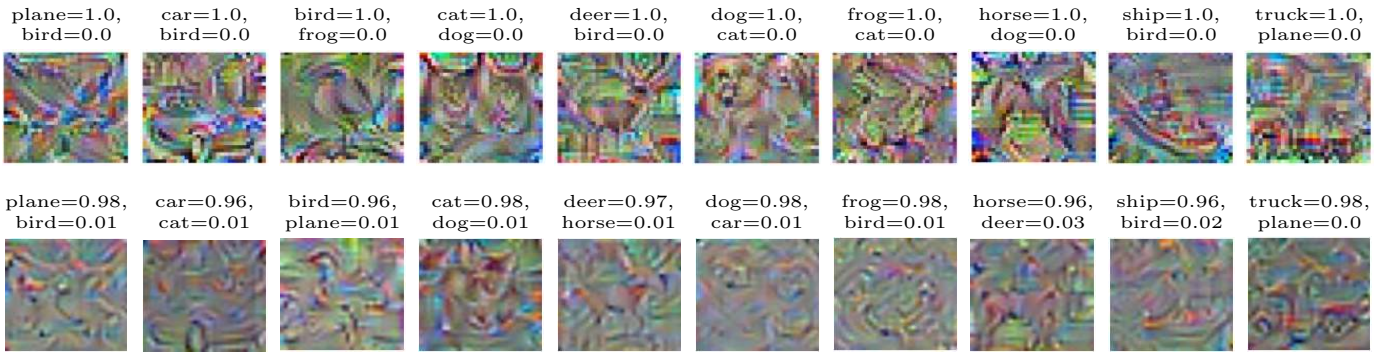

Fig. 2. Visualizing the DIs synthesized from the Teacher model trained on the CIFAR-10 dataset for different choices of output softmax vectors (i.e., output class probabilities). Note that the figure shows 2 DIs per class in each column, each having a different spread over the labels. However, only the top-2 confidences in the sampled softmax corresponding to each DI are mentioned on top for clarity.

becomes a hyper-parameter for the algorithm. Thus the actual sampling of the probability vectors happen from $p(\boldsymbol{s}) = Dir(K, \beta \times \boldsymbol{\alpha})$. β intuitively models the spread of the Dirichlet distribution and acts as a scaling parameter atop $\boldsymbol{\alpha}$ to yield the final concentration parameter (prior). β controls the l_1-norm of the final concentration parameter which, in turn, is inversely related to the variance of the distribution. Variance of the sampled simplexes is high for smaller values of β . However very low values for β (e.g. 0.01), in conjunction with the chosen $\boldsymbol{\alpha}$, result in highly sparse softmax vectors concentrated on the extreme corners of the simplex, which is equivalent to generating class impressions (see Fig. 4). As per the ablation studies, β values of 0.1, 1.0 or a mix of these are in general favorable since they encourage higher diversity (variance) and at the same time does not result in highly sparse vectors. Our proposed approach for generating Data Impressions from a *Trained* classifier is presented in Algorithm 1.

Some of the resulting DIs are presented in Figure 2 for the CIFAR-10 dataset. Note that the figures show 2 DIs per category. Also, note that the top-2 confidences in the sampled softmax corresponding to each DI are mentioned on top. We observe that the DIs are visually far away from the actual data samples of the dataset. However, some of the DIs synthesized from peaky softmax vectors (e.g. the bird, cat, car, and deer in the first row) contain clearly visible patterns of the corresponding objects. The observation of the DIs being visually far away from the actual data samples is understandable, since the objective to synthesize them (eq. (2)) pays no explicit attention to visual detail.

4 Applications of Data Impressions and Experimental Evaluation

The generated Data Impressions through the proposed approach can be utilized for several applications in the absence of training data, which we discuss in detail. We specifically study the application of Data Impression for multiple important CV/ML tasks, viz., Zero-shot knowledge distillation, Unsupervised Domain Adaptation, Continual Learning and Data-free UAP Generation. Here, for each application area, we first introduce the problem and describe how the extracted data-impressions can be utilized towards these tasks. Subsequently, we provide a detailed experimental evaluation to justify the utility of DIs in the given task.

4.1 Zero-Shot Knowledge Distillation

Transferring the generalization ability of a large, complex Teacher (T) deep neural network to a relatively simpler Student (S) network can be achieved using the class probabilities produced by a Teacher as "soft targets" [9] for training the Student. For this transfer, most of the existing approaches require access to the original training data consisting of tuples of input data and targets $(x, y) \in \mathbb{D}$. Let T be the Teacher network with learned parameters θ_T and S be the Student with parameters θ_S, note that in general $|\theta_S| \ll |\theta_T|$. Knowledge distillation methods train the Student via minimizing the following objective (L) with respect to the parameters θ_S over the training samples $(x, y) \in \mathbb{D}$

$$L = \sum_{(x,y)\in\mathbb{D}} L_{KD}(S(x, \theta_S, \tau), T(x, \theta_T, \tau)) + \lambda L_{CE}(\hat{y}_S, y) \tag{3}$$

L_{CE} is the cross-entropy loss computed on the labels $\hat{y}_S$ predicted by the Student and their corresponding ground truth labels y. L_{KD} is the distillation loss (e.g. cross-entropy or mean square error) comparing the soft labels (softmax outputs) predicted by the Student against the soft labels predicted by the Teacher. $T(x, \theta_T)$ represents the softmax output of the Teacher and $S(x, \theta_S)$ denotes the softmax output of the Student. Note that, unless it is mentioned, we use a softmax temperature of 1. If we use a temperature value (τ) different from 1, we represent it as $S(x, \theta_S, \tau)$ and $T(x, \theta_T, \tau)$ for the remainder of the paper. λ is the hyper-parameter to balance the two objectives.

Once we craft the Data Impressions (DI) $(\bar{X})$ from the Teacher model using Algorithm 1, we treat them as the 'Transfer set' and perform the knowledge distillation. Note that we use only the distillation loss L_{KD} as shown in eq. (4). We ignore the cross-entropy loss from the general Distillation objective (eq. (3)) since there is only minor to no improvement in the performance and it reduces the burden of hyper-parameter λ.

$$\theta_S = \underset{\theta_S}{\operatorname{argmin}} \sum_{\bar{x}\in\bar{X}} L_{KD}(T(\bar{x}, \theta_T, \tau), S(\bar{x}, \theta_S, \tau)) \tag{4}$$

Thus we generate a diverse set of pseudo training examples that can provide with enough information to train the Student model via Dirichlet sampling. In the subsequent sections, we discuss the experimental evaluation of the proposed data-free knowledge transfer framework over a set of benchmark object recognition datasets

4.1.1 Experimental Setup and Datasets

We experimentally evaluate our proposed Zero-Shot Knowledge Distillation (ZSKD) approach on MNIST [1], Fashion MNIST (FMNIST) [39], and CIFAR-10 [2]. Here, we provide detailed experimental setup for each of these three datasets.

MNIST: We consider Lenet-5 for the Teacher model and Lenet-5-Half for Student model similar to [11]. The Lenet-5 Model contains 2 convolution layers and pooling which is followed by three fully connected layers. Lenet-5 is modified to make Lenet-5-Half by taking half the number of filters in each of the convolutional layers. The Teacher and Student models have 61706 and 35820 parameters respectively. Input images are resized from 28×28 to 32×32 and the pixel values are normalized to be in $[0, 1]$ before feeding into the models.

Fashion-MNIST: Similar to MNIST, we consider Lenet-5 and Lenet-5-Half as Teacher and Student model respectively where each input image is resized from dimension 28×28 to 32×32.

CIFAR-10: Unlike MNIST and Fashion MNIST, this dataset contains RGB images of dimension $32 \times 32 \times 3$. We take AlexNet [40] as Teacher model which is relatively large in comparison to LeNet-5. Since the standard AlexNet model is designed to process input of dimension $227 \times 227 \times 3$, we need to resize the input image to this large dimension. To avoid that, we have modified the standard AlexNet to accept $32 \times 32 \times 3$ input images. The modified AlexNet contains 5 convolution layers with BatchNorm [41] regularization. Pooling is also applied on convolution layers $1, 2,$ and 5. The deepest three layers are fully connected. AlexNet-Half is derived from this AlexNet by taking half of convolutional filters and half of the neurons in the fully connected layers except in the classification layer which has number of neurons equal to number of classes. The AlexNet-Half architecture is used as the Student model. The Teacher and Student models have 1.65×10^6 and 7.23×10^5 parameters respectively.

4.1.2 Implementation Details

As all the experiments in these three datasets are dealing with classification problems with 10 categories each, value of the parameter K in all our experiments is 10. For each dataset, we first train the Teacher model over the available training data using the cross-entropy loss. Then we extract a set of Data Impressions (DI) from it via modelling its softmax output space as explained in sections 3.1 and 3.2. Finally, we choose a (light weight) Student model and train over the transfer set (DI) using eq. (4).

We consider two $(B = 2)$ scaling factors, $\beta_1 = 1.0$ and $\beta_2 = 0.1$ across all the datasets, i.e., for each dataset, half the Data Impressions are generated with β_1 and the other with β_2. However we observed that one can get a fairly decent performance with a choice of beta equal to either 0.1 or 1 (even without using the mixture of Dirichlet) across the datasets. A temperature value (τ) of 20 is used across all the datasets. Also, since the proposed approach aims to achieve better generalization, it is a natural choice to augment the crafted Data Impressions while performing the distillation. We augment the samples using regular operations such as scaling, translation, rotation, flipping etc. which has proven useful in further boosting the model performance [42].

In section 4.1.3, we show the ZSKD results on the three benchmark datasets. In the subsequent sections, we investigate in detail, the effect of transfer set size, i.e., the number of Data Impressions on the performance of the Student model (sec. 4.1.4), compare the ZSKD results when used with Class Impressions [35] (sec. 4.1.5), apply ZSKD on large architectures (sec. 4.1.6) and finally show that DIs preserve adversarial robustness in the ZSKD framework (sec. 4.1.7).

4.1.3 Results and Discussion

The performance of Zero-Shot Knowledge Distillation for the MNIST, Fashion-MNIST, and CIFAR-10 datasets is presented in Tables 1, 2, and 3 respectively. Note, that in order to understand the effectiveness of the proposed ZSKD, the tables also show the performance of the Teacher and Student models trained over actual data samples along with a comparison against existing distillation approaches. Teacher-CE denotes the classification accuracy of the Teacher model trained using the cross-entropy (CE) loss, Student-CE denotes the performance of the Student model trained with all the training samples and their ground truth labels using cross-entropy loss. Student-KD denotes the accuracy of the Student model trained using the actual training samples through Knowledge Distillation (KD) from Teacher. Note that this result may act as an upper bound for the data-free distillation approaches.

TABLE 1
Performance of the proposed ZSKD framework on the MNIST dataset.

Model	Performance
Teacher-CE	99.34
Student-CE	98.92
Student-KD [9] 60K original data	99.25
[43] 200 original data	86.70
[11] (uses meta data)	92.47
ZSKD (Ours) (24000 DIs, and no original data)	98.77

Table 1 presents our results on MNIST, and compares them with existing approaches. It is clear that the proposed Zero-Shot Knowledge Distillation (ZSKD) outperforms the existing few data [43] and data-free counterparts [11] by a great margin. Also, it performs close to the full data (classical) Knowledge Distillation while using only 24000 *DI*s, i.e., 40% of the the original training set size.

Table 2 presents our results for Fashion-MNIST and compares them with the existing approaches. Similar to MNIST, ZSKD outperforms the existing few data knowledge distillation approach [43] by a large margin, and performs close to the classical knowledge distillation scenario [9] with all the training samples.

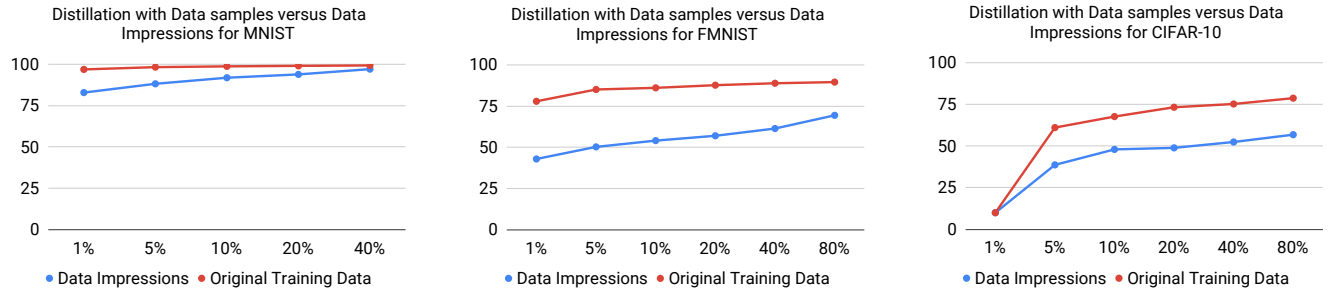

Fig. 3. Performance (Test Accuracy) comparison of Data samples versus Data Impressions (without augmentation). Note that the x-axis denotes the number of DIs or original training samples (in %) used for performing Knowledge Distillation with respect to the size of the training data.

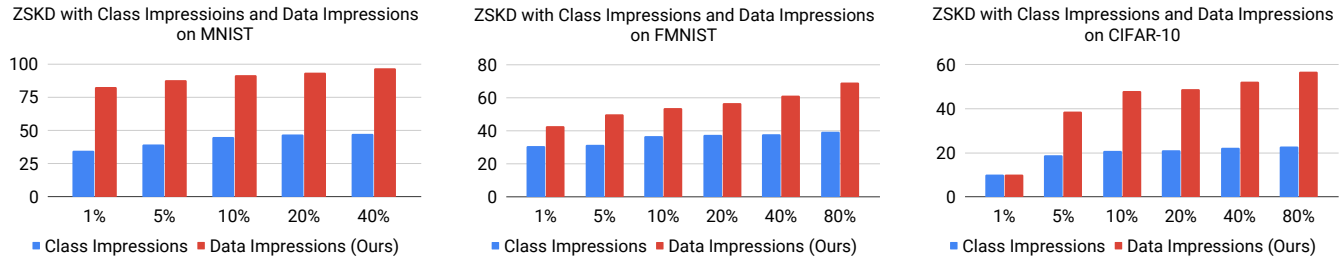

Fig. 4. Performance (Test Accuracy) comparison of the ZSKD with Class Impressions [35] and proposed Data Impressions (without augmentation). Note that the x-axis denotes the number of DIs or CIs (in %) used for performing Knowledge Distillation with respect to the training data size.

TABLE 2
Performance of the proposed ZSKD framework on the Fashion MNIST dataset.

Model	Performance
Teacher-CE	90.84
Student-CE	89.43
Student-KD [9] 60K original data	89.66
[43] 200 original data	72.50
ZSKD (Ours) (48000 DIs, and no original data)	79.62

TABLE 3
Performance of the proposed ZSKD framework on the CIFAR-10 dataset.

Model	Performance
Teacher-CE	83.03
Student-CE	80.04
Student-KD [9] 50K original data	80.08
ZSKD (Ours) (40000 DIs, and no original data)	69.56

Table 3 presents the results on CIFAR-10 dataset. It can be observed that the proposed ZSKD approach can achieve knowledge distillation with the Data Impressions that results in performance competitive to that realized using the actual data samples. Since the underlying target dataset is relatively more complex, we use a bigger transfer set containing 40000 DIs. However, the size of this transfer set containing DIs is still 20% smaller than that of the original training set size used for the classical knowledge distillation [9].

4.1.4 *Effect of Transfer Set Size*

In this subsection, we investigate the effect of transfer set size on the performance of the distilled Student model. We perform the distillation with different number of Data Impressions such as $\{1\%, 5\%, 10\%, \ldots, 80\%\}$ of the training set size. Figure 3 shows the performance of the resulting Student model on the test set for all the datasets. For comparison, the plots present performance of the models distilled with the equal number of actual training samples from the dataset. It is observed that, as one can expect, the performance increases with size of the transfer set. Also, note that the initial performance (with smaller transfer set) reflects the complexity of the task (dataset). For simpler datasets such as MNIST, smaller transfer sets are sufficient to achieve competitive performance. In other words, small number of Data Impressions can do the job of representing the patterns in the dataset. As the dataset becomes complex, more number of Data Impressions need to be generated to capture the underlying patterns in the dataset. Note that similar trends are observed in the distillation with the actual training samples as well.

4.1.5 *Class Versus Data Impressions*

Feature visualization works such as [7], [8], [44], [45] attempt to understand the patterns learned by the deep neural networks in order to recognize the objects. These works reconstruct a chosen neural activation in the input space as one way to explain away the model's inference.

As described earlier, one of the recent works by [35] reconstructs samples of a given class for a downstream task of adversarial fooling. A random noise is optimized in the input space till it results in a one-hot vector (softmax) output. This means, their optimization to craft the

TABLE 4
Performance measures to evaluate the robustness transfered under distillation using Data Impressions for different datasets. A_{nat} denotes the accuracy obtained on unperturbed data whereas A_{adv} denotes adversarial accuracy i.e. the performance of the model on the perturbed data. F.R. is the 'fooling rate' which describes the amount of samples whose labels got changed after adversarial attack. All the numbers shown are in %.

Dataset	Model	Performance Measures (in %)						
		Natural	FGSM		IFGSM		PGD	
		$A_{nat}\uparrow$	$A_{adv}\uparrow$	F.R. $\downarrow$	$A_{adv}\uparrow$	F.R. $\downarrow$	$A_{adv}\uparrow$	F.R. $\downarrow$
MNIST	Non-robust teacher	99.34	21.01	79.45	1.65	98.87	0.51	99.97
	Student using DIs from non-robust teacher	98.77	37.4	63.44	8.52	92.39	3.06	97.86
	Robust teacher	98.01	97.06	2.24	95.04	5.66	95.0	5.6
	Student using DIs from robust teacher	95.85	86.1	13.4	74.01	28.44	54.38	48.27
Fashion-MNIST	Non-robust teacher	90.84	13.99	91.47	6.02	99.99	4.85	100.0
	Student using DIs from non-robust teacher	79.62	13.94	96.26	12.03	100.0	9.64	100.0
	Robust teacher	75.15	74.28	7.77	72.92	13.5	73.41	12.83
	Student using DIs from robust teacher	68.44	60.25	31.80	47.90	58.78	39.81	70.14
CIFAR-10	Non-robust teacher	83.03	15.89	93.46	10.16	99.31	9.62	99.87
	Student using DIs from non-robust teacher	69.56	17.74	97.52	15.67	99.74	15.24	99.86
	Robust teacher	66.99	51.72	53.24	50.56	55.23	46.53	60.86
	Student using DIs from robust teacher	54.38	37.44	70.89	32.45	77.99	24.92	86.62

representative samples would expect a one-hot vector in the output space. Hence, they call the reconstructions Class Impressions. Our reconstruction (eq. (2)) is inspired from this, though we model the output space utilizing the class similarities perceived by the Teacher model. Because of this, we argue that our modelling is closer to the original distribution and results in better patterns in the reconstructions, calling them Data Impressions of the Teacher model.

We compare these two varieties of reconstructions for the application of distillation. Figure 4 demonstrates the effectiveness of Class and Data Impressions over three datasets. It is observed that the proposed Dirichlet modelling of the output space and the reconstructed impressions consistently outperform their class counterparts by a large margin. Also, in case of Class Impressions, the increment in the performance due to increased transfer set size is relatively small compared to that of Data Impressions. Note that for better understanding, the results are shown without any data augmentation while conducting the distillation.

4.1.6 Performance of ZSKD on Large Architectures

In this section, we investigate the performance of ZSKD on popular network architectures, in addition to those studied in Sec. 4.1.3. Note that, these architectures are also of much larger capacity than that of the models discussed earlier. Specifically, we perform experiments on VGG and Resnet architectures on the CIFAR-10 dataset.

TABLE 5
ZSKD performance using data impressions from VGG Teacher architecture on CIFAR-10

Model	Data-free	Performance (%)
VGG-19 (T)	✗	87.99
VGG-11 (S)- CE	✗	84.19
VGG-11 (S)- KD [9]	✗	84.93
VGG-11 (S)- ZSKD (**Ours**)	✓	74.10
Resnet-18 (S)- CE	✗	84.45
Resnet-18 (S)- KD [9]	✗	86.58
Resnet-18 (S)- ZSKD (**Ours**)	✓	74.76

As shown in Table 5, VGG-19 is taken as the Teacher network which is trained for 500 epochs with a learning rate (lr) of 0.001 and batch size of 512. The knowledge from the trained Teacher is distilled into two different student models i.e. VGG-11 and Resnet-18. Their performance on original training data without (CE) and with distillation (KD) are also reported (the latter can be assumed as an upper bound). The data impressions are generated using adam optimizer with a batch size of 32 and initial learning rate of 10 with a β mixture of $\{0.1, 1.0\}$. The learning rate is subsequently reduced linearly over the 1500 iterations of optimization. The ZSKD performance on VGG-11 and Resnet-18 while distilling from the VGG-19 teacher with a learning rate of 0.001 are 74.10% and 74.76% respectively.

TABLE 6
ZSKD performance using Data Impressions from Resnet-18 Teacher architecture for CIFAR-10

Model	Data-free	Performance (%)
Resnet-18 (T)	✗	86.54
Resnet-18-half (S)- CE	✗	85.51
Resnet-18-half (S)- KD [9]	✗	86.31
Resnet-18-half (S)- ZSKD (**Ours**)	✓	81.10

We also perform the experiments with a different Teacher network architecture i.e. Resnet-18 which is trained with lr 0.01, batch size of 512 for 500 epochs and obtain an accuracy of 86.54%. Here, we use Resnet-18-half as a student network which is formed by taking half the number of filters at each layer of Resnet-18. Similar to the previous experiment, we also report results with and without distillation using original training data as shown in Table 6. The data impressions are synthesized with a lr of 0.001. Our ZSKD method obtains an accuracy of 81.10% which is only $\approx 5\%$ less than the performance using the entire original training data (KD).

4.1.7 Investigating Adversarial Robustness of DI-Distilled models

In this subsection, we demonstrate that Data Impressions are indeed close-to-true approximations of the training data by experimentally verifying that they capture the adversarial robustness property of an adversarially trained Teacher, and preserve it under zero-shot knowledge distillation.

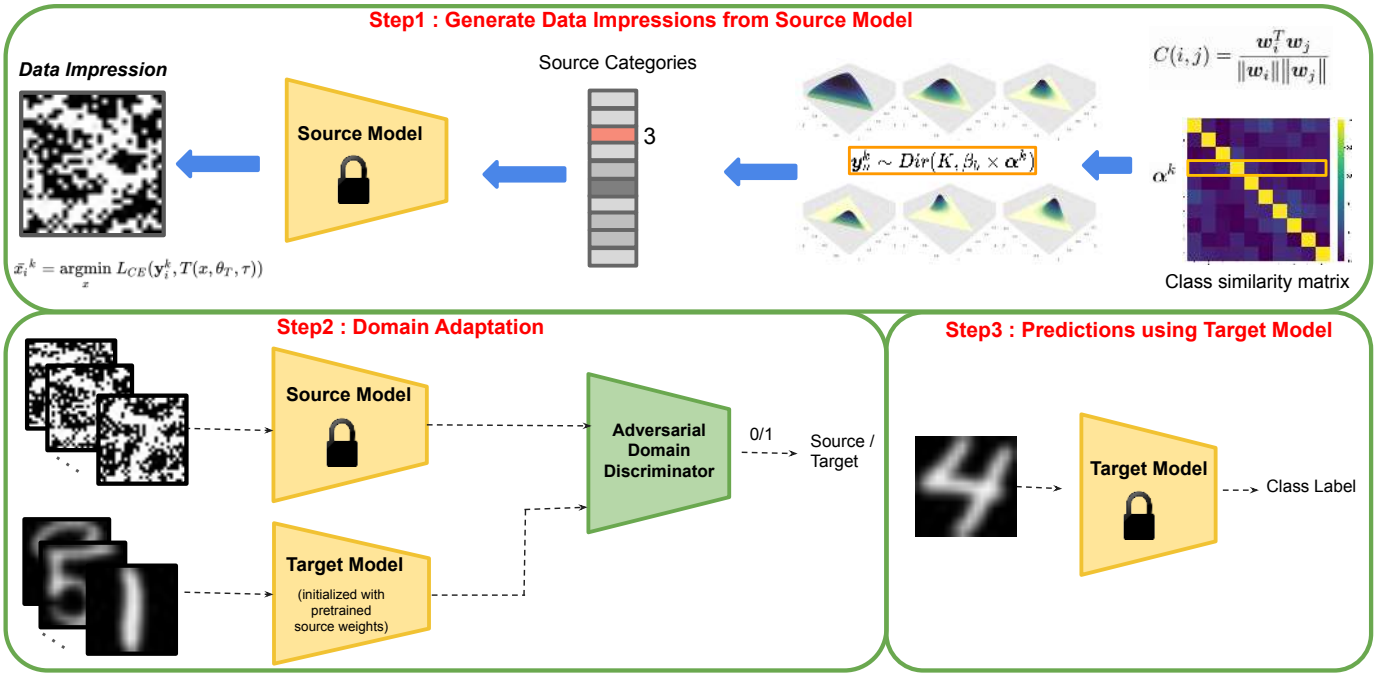

Fig. 5. Proposed Approach for Source Free Unsupervised Domain Adaptation using Data Impressions.

While multiple works such as [12], [13] have studied data-free approaches for training deep neural networks, to the best of our knowledge, we are the first to study the robustness properties of the resulting Student models. We empirically analysed the behaviour of Students that are distilled from normally trained versus adversarially trained Teachers. The distribution of adversarial samples (generated by perturbing natural images) would likely be different from the natural training data distribution. Therefore, it is critical to study if Data Impressions capture enough information about a robust Teacher to pass this property on to smaller Students.

We posit that since adversarially trained networks are better-equipped to approximate the posterior probabilities over the adversarially perturbed data [46], the *Data Impression* generating process is able to draw samples from the perturbed training distribution. In other words, the produced *Data Impressions* behave as surrogates to the perturbed training data, which when used for distillation, allow the Student to also be adversarially robust.

To demonstrate this, we craft Data Impressions from adversarially-trained Teachers by exactly following the methodology described in Section 3. Without enforcing explicit regularization or any additional penalty, we are able to produce robust Student networks under knowledge distillation in the data-free scenario.

In Table 4, we experimentally compare the performance of Student networks distilled from Data Impressions crafted from both naturally-trained and adversarially robust Teacher networks when subjected to commonly used adversarial attacks, viz., FGSM [47], iFGSM [48], PGD [46]. The Teacher networks (as described in Section 4.1.2 for MNIST, F-MNIST, CIFAR-10) are made robust through PGD adversarial training [46]. While, it is interesting to note that the Students distilled through ZSKD from non-robust Teachers show slightly improved adversarial accuracies than the Teachers themselves, the students are not completely robust. In the case of robust Teachers however, significant robustness is passed down to the Student networks.

In subsequent sections, we present other applications to demonstrate the general applicability of Data Impressions as a surrogate to the true training data distribution when the latter is unavailable.

4.2 Domain Adaptation

In this section, we demonstrate the applicability of Data Impressions for the task of unsupervised closed set Domain Adaptation.

A model trained on data samples from a source distribution often does not generalize well when it encounters samples from a different target distribution due to domain gap or the dataset bias. In cases where the target data is unlabelled, possibility of finetuning the source model on target dataset becomes impractical. In order to reduce this domain shift, unsupervised domain adaptation techniques have gained a lot of attention recently. Based on the overlap between source and target label sets, there are different categories of domain adaptation: closed set, partial, open set and universal [49]. We restrict our discussion to closed set domain adaptation where the labels are shared between source and target domains.

During the deployment of source model, the source data that has been used for training may not be available due to several reasons such as data privacy, proprietary rights over the data, cost associated with sharing a large dataset etc. (also explained in section 1). However, most of the existing works depend on the availability of both the source and target data for domain adaptation (also discussed in unsupervised domain adaptation paragraph of section 2). We overcome this limitation by generating Data Impressions

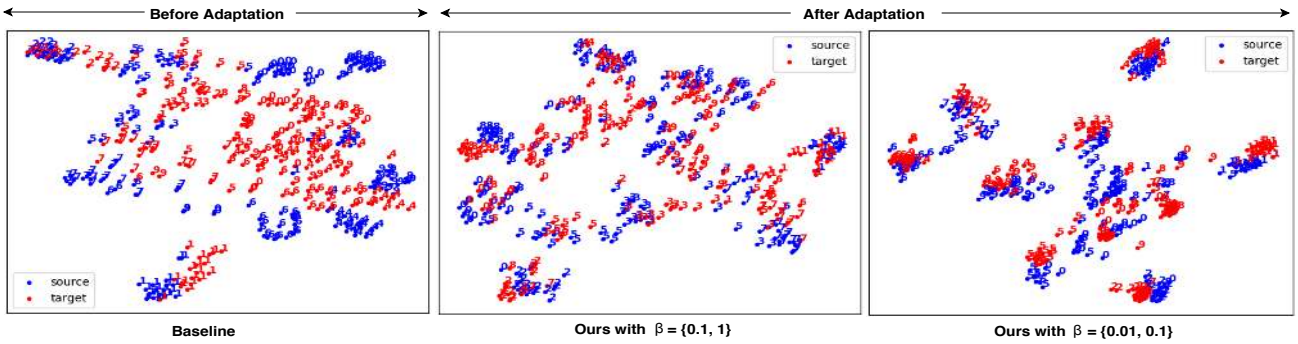

Fig. 6. TSNE Plots to visualize the source free domain adaptation of USPS → MNIST through our proposed approach via Data Impressions

from the source model that act as a proxy to the source data. Thus, Data Impressions enable any relevant domain adaptation technique to be easily adopted for source free domain adaptation task.

4.2.1 Experimental Setup and Datasets

We perform experiments on SVHN [50], MNIST [1] and USPS [51] where we adapt:

SVHN to MNIST: In order to have a fair comparison with other works, the entire training data is used for this adaptation experiment.

MNIST to USPS: We use the training protocol followed in [52] where 2000 and 1800 images are sampled from MNIST and USPS respectively.

USPS to MNIST: We use the same training protocol as followed in the MNIST to USPS experiment.

All the training data are resized to 28×28 and pixel values are normalized between 0 and 1. We use the same LeNet architecture as described in [25] for all the domain adaptation experiments. Note that the target data labels are not used while training.

4.2.2 Implementation

We use a popular unsupervised domain adaptation technique by Tzeng *et al.* [25] as a backbone, owing to its effectiveness and simplicity. We use their implementation [53] to get the baseline performances. Overview of our proposed method is shown in Figure 5. In step 1, the Data Impressions are generated from the pretrained source network using Algorithm 1. In the second step, the pretrained source model is frozen and the parameters of the target model are learned. The target model is initialized with weights of pretrained source network. The input to the source and target models are Data Impressions and unlabeled target data respectively. The outputs of the source and target models are then fed to an adversarial domain discriminator, which is trained with the objective of correctly identifying the domains of the inputs. The discriminator has two fully connected layers of 500 neurons each with leaky ReLU as activation function and the final layer yields two outputs. The target model, however, is trained to confuse the discriminator using the adversarial loss. Finally, step 3 performs the inference, where the trained target model is evaluated on the target data.

TABLE 7
Comparison with source dependent domain adaptation works. Notations- S:SVHN, M:MNIST, U:USPS

Method (Source → Target)	**S → M**	**U → M**	**M → U**
Gradient Reversal [54]	0.739	0.730	0.771
Domain Confusion [55]	0.681	0.665	0.791
CoGAN [56]	-	0.891	**0.912**
ADDA [25]	0.76	**0.901**	0.894
Baseline	0.612	0.573	0.765
Ours (Source-free) ($\beta = \{0.01, 0.1\}$)	**0.866**	0.8915	0.91

4.2.3 Results and Discussion

Results are presented in Table 7. The baseline performance represents directly utilizing the source model (without domain adaptation) to predict the labels for the target data. In our experiments with multiple choices for mixtures of β, we have typically observed that with lower β values we achieve better performance. For example, with a mixture of $\beta = \{0.1, 1.0\}$, we achieve substantially better than the baseline results. However, it can be observed from Table 7 that β when taken as mixture of $\{0.01, 0.1\}$ gives the best results across all the datasets. This is in line with the fact that *lower β values encourage more class specific Dirichlet softmax vectors to be sampled* (section 3.2).

In order to better understand, we use TSNE plots for visualization in Figure 6, where USPS is adapted to MNIST. We can observe that before adaptation, the source and target data are not aligned. After adaptation using Data Impressions, the source and target data starts getting aligned. With proper mix of β values, the target data samples are well separated and the data clusters become more compact and tight.

We further compare our proposed approach with other works that use source data as shown in Table 7. It can be easily observed that domain adaptation using Data Impressions gives competitive or better domain performance over several source dependent techniques.

4.2.4 Comparison with Recent Source-free Domain Adaptation methods

In this section, we compare our results on several datasets against some of the recent source-free domain adaptation

works (refer section 2 for the method comparison).

TABLE 8
Comparison with SDDA. Notations- S:SVHN, M:MNIST, U:USPS

Method	Source baseline	Adaptation	Improvement over baseline
SDDA (S → M)	67.2	76.3	9.1
Ours (S → M)	61.74	93.55	**31.81**
SDDA (M → U)	82.5	88.5	6.0
Ours (M → U)	83.81	94.56	**10.75**

Kurmi *et al.* [31] proposed *'Source Data free Domain Adaptation' (SDDA)* method to handle the unavailability of source data during unsupervised domain adaptation. We compare our performance on their source network architecture on two different adaptations : SVHN to MNIST and MNIST to USPS. As per their protocol, we use the entire training data of the datasets for adaptation, unlike previous experiments where only 1800 and 2000 training images of USPS and MNIST were used. For both the adaptations, we train the network on the source data with learning rate 0.001 and adam optimizer. The data impressions are generated with learning rate 0.001. We use a β mixture of 0.01 and 0.1 during generation of data impressions. The adaptation with generated impressions are performed with learning rate $2e^{-4}$ and adam optimizer. The results obtained are compared with SDDA as shown in Table 8. Our method performs significantly better and achieves a large improvement of 31.81% and 10.75% over baseline while performing adaptation from SVHN to MNIST and MNIST to USPS respectively.

TABLE 9
Comparison with SHOT. Notations- S:SVHN, M:MNIST, U:USPS

Method	Source baseline	Adaptation	Improvement over baseline
SHOT (S → M)	65.72	89.62	23.9
Ours (S → M)	61.20	86.6	**25.4**
SHOT (U → M)	58.55	88.65	30.1
Ours (U → M)	57.30	89.15	**31.85**
SHOT (M → U)	76.06	85.83	9.77
Ours (M → U)	76.50	91.0	**14.5**

Liang *et al.* [30] proposed *'Source HypOthesis Transfer' (SHOT)* which uses different source network architectures for adaptation for MNIST ↔ USPS and SVHN → MNIST. Moreover, their proposed networks are customized with addition of batchnorm layers and weight normalization layers at the end of the feature extraction module and classifier module respectively. In order to have a fair comparison with ours, we make some modifications to the SHOT pipeline. Specifically, we replace their architectural dependent source network with our network and the "smooth loss function" used in their method is replaced with traditional cross entropy, as used for training our network. Similar to ours, we use 1800 and 2000 images sampled from USPS and MNIST respectively while performing adaptation of the classifier from USPS to MNIST and MNIST to USPS. Also, we use the same data preprocessing as used in ours i.e. normalizing each input pixel between 0 to 1. The adaptation performance achieved by SHOT on these aforementioned modifications is compared vis-a-vis ours in Table 9. As evident, our improvement in performance over baseline is better on adaptations across different datasets.

In Tables 8 and 9, the difference in the source baseline performances between ours and compared methods, can be attributed to the chosen hyperparameters such as initial learning rate, number of epochs, learning rate scheduler, etc. used for training the source network. The performance of SDDA mentioned in Table 8 for different adaptations (SVHN → MNIST and MNIST → USPS) are the numbers reported from their paper. However, the weights of their pretrained source network were not available. Thus, we trained their source network architecture and performed our adaptation on it. Nevertheless, in order to have a fair comparison and to discount the performance difference in the baseline, we compare the improvement in performance over the baseline (i.e. difference between the method's domain adaptation and its source baseline performance) between ours and SDDA methods. Similarly, in Table 9, we reported the performance of SHOT on our architecture and then compared our performance. We used the default hyperparameters of the official github repository of SHOT while training the source network on our architecture. That resulted in better source baseline performance of SHOT (SVHN → MNIST) as compared to ours. However, it is evident from both the Tables that we obtain more improvement in performance over the source baselines which demonstrates the efficacy of our proposed adaptation technique.

4.3 Continual Learning

In this section, we present the application of Data Impressions for the task of continual learning. There are several flavours of continual learning such as class incremental learning, domain incremental learning and task incremental learning [57]. We demonstrate the usability of Data Impressions for incrementally learning objects from different classes. In this setting, the knowlede obtained by neural network model from old classes is compromised while trying to learn from new classes. The exemplars from old classes cannot be stored due to the implicit assumption of limited memory constraints. In order to have a fair comparison, we restrict our discussion with works which do not use any exemplars from old classes. Therefore we do not consider works such as [19], [20] that store exemplars which are carefully selected to avoid catastrophic forgetting.

Since the training data that belongs to old classes is not available, some simple baselines can be adopted such as finetuning and fixed representation. In the former case, the model which is previously trained on old classes is finetuned with labelled samples of new classes while in the latter case, the model is frozen and only the last layer are trained that are connected to the new class labels. LwF [22] is an important baseline that we compare against. They utilize samples from new categories for minimizing (i) the distillation loss on the old classes in order to avoid catastrophic forgetting, and (ii) cross entropy loss on the new classes. We also do comparison of our proposed method with another recent method named *Deep Model Consolidation* (DMC) by Zhang *et al.* [23]) which utilized publicly available auxiliary data for class incremental learning in the absence of exemplars. Our method synthesizes data impressions

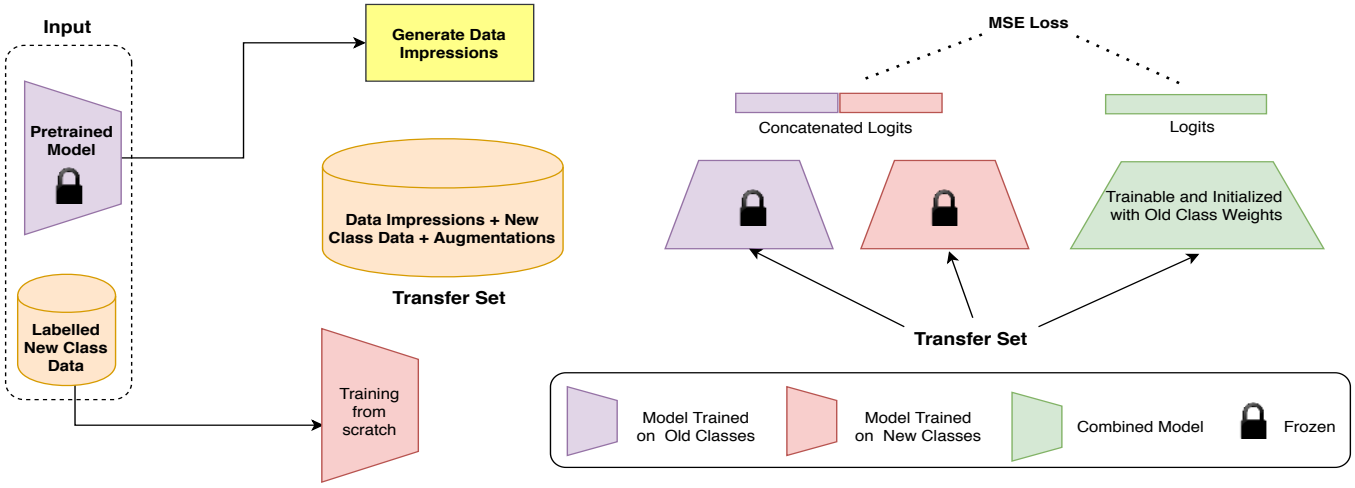

Fig. 7. Proposed Approach for Continual Learning using Data Impressions in the absence of old class data.

using the model trained on old classes, which are then used as a substitute to the samples belonging to old categories. Hence, unlike [23]) our proposed approach for continual learning do not require any arbitrary data.

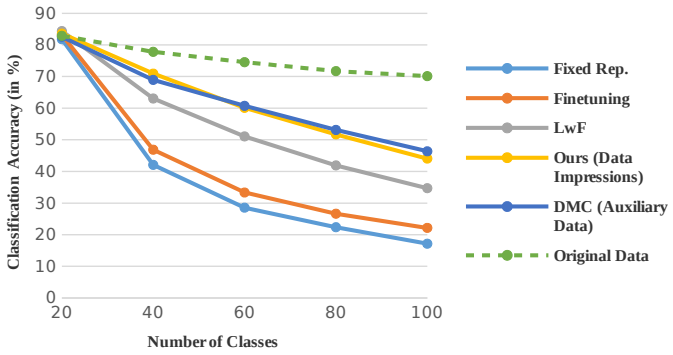

Fig. 8. Performance comparison of incremental learning experiments on CIFAR-100 dataset with a step size of 20 classes.

4.3.1 Experimental Setup and Datasets

The experiments are performed on CIFAR-100 dataset [2] with an incremental step of 20 classes. The data is normalized with channel mean and standard deviation of 0.5, then the normalized data is fed as an input to the model. In order to have a fair comparison, we use the same model architecture as in [19], [20], [22], [23] i.e., the ResNet-32 [58]. In this exemplar-free setup, for each subsequent incremental step, apart from the samples of new classes, we only have access to the model weights trained on the old classes, but not the old class data samples themselves.

4.3.2 Implementation

The proposed approach is shown in Figure 7. Since we consider a limited memory scenario, we generate only 2400 Data Impressions overall. As the count of old classes increases after few incremental steps, the number of Data Impressions generated per class decreases and hence representing old classes with less generated data is challenging. Therefore, we perform simple augmentations such as flipping, rotations, scaling etc. on the generated data impressions. The dual distillation loss [23] is used for training the combined model. Note that unlike [23], we do not use any auxiliary data, instead the generated Data Impressions and labelled samples of new classes are used as a transfer set. Also, while training the combined model, we initialize with the weights of old class model as it results in better performance compared to training from scratch.

When we independently train the model on new classes data, we use an initial learning rate of 0.1. The combined model is trained with an initial learning rate of 0.01 for all the incremental steps except for the last incremental step where we use a learning rate of 0.001. Across all the incremental experiments, we use SGD optimizer with momentum of 0.9 and weight decay of 0.0005. The learning rate is reduced by $1/5$ after every 70 epochs and training is done for a maximum of 500 epochs.

4.3.3 Results and Discussion

The results are shown in Figure 8 where the mean accuracy of 5 trials is reported. We perform significantly better than LwF [22] at every incremental step and close to DMC [23] (which uses additional auxiliary data). The incremental learning performance using all the samples of original data are also shown through dashed lines which serves as an upper bound.

The Fixed Representation and Finetuning baselines have severe limitations. Both of these approaches either perform well on old classes or on new classes but not on both. In the exemplar-free incremental learning setup, the challenge is to balance the performance for both the old and new classes. However, in the Fixed Representation approach, the model does not have enough capacity to learn the new classes very well and so its performance ends up being biased towards the old classes. On the other hand, in the Finetuning approach, the entire model is updated for the new classes, and so the performance is biased towards the new classes. In our approach, we generate and utilize DIs as pseudo-exemplars from the old class data and use it in conjunction with the data samples from the new classes in each incremental step. This enables achieving a nice balance

in performance across both the old and new classes, as evidenced by a major improvement in performance over the aforementioned methods (see Fig. 8).

Our method reports a performance very similar to DMC. However, DMC carries its own set of limitations: It utilizes auxiliary data in the absence of exemplars. Availability of such data is a strong assumption, especially considering our strictly "data-free" experimental setting. Infact, we may not have the luxury of such unlabelled data in several specialized domains such as medical imaging, satellite/aerial imaging, etc. Furthermore, DMC's performance is dependent on how close the choice of the auxiliary data is to the original training data. Our approach overcomes these limitations by using *Data Impressions* as surrogates to the old classes which makes our method independent of any additional auxiliary data.

4.4 Universal Adversarial Perturbations

In this section, we demonstrate the use of Data Impressions to craft Universal Adversarial Perturbations (UAPs) [32]. These perturbations are input-agnostic imperceptible noises that, when added to the natural data samples, can fool a target classifier into misclassifying them.

UAPs are typically powerful attacks even in the black-box setting, and it is critical to study them, especially as they have been shown to be effective in the data-free scenario. Mopuri *et al.* [35] realize data-free UAPs by training a generative model using Class Impressions. We craft UAPs by utilizing Data Impressions, and compare the results in Table 10.

4.4.1 Experimental Setup and Datasets

We use the Data Impressions obtained from the LeNet and AlexNet classifiers described in sections 4.1.1 (for MNIST), 4.1.1 (for FMNIST), and 4.1.1 (for CIFAR-10) respectively.

We use a generator architecture modified from [59] for a 32×32 input, consisting of 4 deconvolutional layers, to generate the UAPs. The final layer is a $tanh$ nonlinearity scaled by ϵ, in order to generate UAPs within the imperceptible ϵ-ball. For a fair comparison, inline with the existing works, an ϵ value of 10 is chosen for imperceptible perturbation in the $[0, 255]$ range, and is scaled accordingly with the signal range of our input.

4.4.2 Implementation

We train the generator that takes a mini-batch of random vectors z sampled from a uniform distribution $U[-1, 1]$ as input and converts them into UAPs through a series of deconvolution layers. The objective for the generator consists of a Fooling Loss and a Diversity Loss, taken from [35] and used in linear combination as described therein.

The generator maps the latent space Z, consisting of 10-dimensional random vectors sampled from $U[-1, 1]$ with a minibatch size of 32, to the UAPs for the target classifier. The architecture remains unchanged for all the experiments, and the generator objective is optimized using Adam. The generator is trained for 20 epochs with a batch size of 32 for each experiment. A hyperparameter α is used to scale the Diversity Loss [35] before adding it to the Fooling Loss. For CIFAR-10, an α of 3e-04, and a learning rate of 1e-05 is used. For both FMNIST and MNIST however, an α of 1e-04 and a learning rate of 1e-05 is used. Figure 9 shows sample UAPs learned by using Data Impressions extracted from target classifier (Alexnet) pretrained on CIFAR-10.

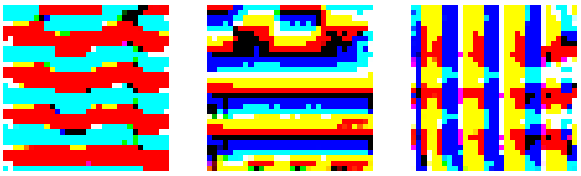

Fig. 9. Visualization of the UAPs crafted from CIFAR-10 Data Impressions.

4.4.3 Results and Discussion

Table 10 compares the fooling rates of the UAPs crafted from Data Impressions and Class Impressions crafted from the same classifiers. It can be observed that the UAPs from Data Impressions achieve better fooling rates and outperform those of Class Impressions by a minimum of 4.05%. Owing to the better representation of the underlying training data by our Data Impressions compared to the Class Impressions [5], the same generative model can now craft better perturbations which is manifested in the form of better fooling rates.

The class impressions are estimated as inputs that maximize the softmax outputs/logit, corresponding to the specific class. Therefore, it is obvious that the CIs are class-specific and the samples generated for each class exhibit very little diversity. On the contrary, the DIs we estimate are not tied to any specific class and are generated for softmax vectors sampled from a Dirichlet distribution with diverse values of the entropy of the target softmax. This leads to the possibility of creating a training set for UAP generation, composed of statistically uncorrelated as well as visually diverse image samples. In fact, the CIs can be shown to be DIs generated for one-hot encoded target softmax vectors, thereby making them just a special case and a small subset of the corresponding set of data impressions. Due to this improvement in quality of the image set, we are able to craft diverse set of strong UAPs leading to better fooling rates.

TABLE 10
Comparison of Fooling Rates (in %) of UAPs crafted from Class Impressions and Data Impressions

Method	AlexNet (CIFAR-10)	LeNet (Fashion-MNIST)	LeNet (MNIST)
CI: AAA [35]	90.18	91.29	91.10
DI: Ours	**94.23**	**96.37**	**96.45**

5 Key Observations

In this section, we summarize our major findings based on the extensive experiments performed with data impressions across multiple different applications.

In ZSKD, β is an important scaling parameter which controls the spread of the Dirichlet distribution. Empirically, we observed better performance when β is a mixture of 1.0 and 0.1. This encourages higher diversity (variance) and at the same time does not result in highly sparse vectors

in comparison to the smaller β mixture of 0.1 and 0.01. Robustness of the Teacher implicitly gets transferred to the Student models through the data impressions, without explicitly adding any extra objective during its generation. Thus, our proposed method for extracting impressions by design itself, closely approximates the data distribution on which the Teacher network is trained.

Another interesting observation is that the student models distilled using data impressions from a non-robust teacher network, obtains slightly higher adversarial accuracy across several adversarial attacks over different datasets in comparison to the performance of corresponding Teacher. This robustness in Student networks can be explained with the fact that the data impressions do not visually 'look' exactly like the training images themselves, but actually only capture the 'essence' of the training data. Thus generated synthetic data impressions are analogous to that of 'adversarial' samples with no bound on perturbation (no ϵ constraint) as the Teacher network classifies them similar to original training samples.

Class impressions [5] can be considered as a special case of data impressions. Small values of $\beta's$ are chosen to enforce the softmax vectors to be sampled from the corners of the simplex in the Dirichlet distribution, making them highly class specific (Proof provided in the supplementary). Based on the experiments performed across multiple datasets, it is evident that the data impressions have clearly outperformed class impressions in both distillation and also in crafting the UAPs. Hence, modelling the softmax space via Dirichlet distribution for extracting surrogate samples is better in comparison to the one-hot vector modelling.

For domain adaptation, the data impressions generated with smaller $\beta's$ (e.g. mixture of 0.1 and 0.01) works better. This shows that the diversity induced in the impressions through high β is not as important as retaining more class information with lower $\beta's$ for this application. In the case of incremental learning, we performed distillation from two separate models trained with old and new classes data respectively into a combined model. We used the transfer set consisting of data impressions and new class data and observed performance as good as the DMC method [23] which assumes access to the auxiliary data. We also observed that the initialization of the combined model with old class weights is better than training the combined model from scratch since the optimization gets easier and leads to better performance.

We choose to show the efficacy of data impressions on some of the most popular applications. We followed the benchmark problem setup and datasets to evaluate the performance of the generated data impressions. Note that these impressions are not generated specifically targeting any particular application, which makes them independent of the target application and hence they can be used in other applications beyond the ones we have demonstrated.

6 Conclusion

In this paper we introduced a novel and interesting problem of restoring training data from a trained deep neural network. Utilizing only the parameters of the trained model but no additional prior to achieve this makes it a challenging task. Hence, we rather focused on a simplified problem. We aimed to restore the training data in a *learning* sense. In other words, our objective is to restore data that can train models on related tasks and generalize well onto the natural data. Apart from the natural academic interest, the presented task has wide practical applicability. Especially it has great value in adapting the laboratory trained deep models into complex data-free scenarios as detailed in section 1. In that regard, we have demonstrated the fidelity of the extracted samples, known as Data Impressions, via realizing excellent generalization for multiple tasks such as Knowledge distillation, crafting Adversarial Perturbations, Incremental Learning, and Domain Adaption. However, one can notice that, although Data Impressions capture some of the striking visual patterns from the actual training data samples, they are visually far away from the training data. Strong priors about the natural training distribution might be needed to improve the visual similarity, an aspect we leave for future investigation.

Acknowledgments

This work is partially supported by 1. Start-up Research Grant (SRG) from SERB, DST, India (Project file number: SRG/2019/001938) and 2. Young Scientist Research Award (Sanction no. 59/20/11/2020-BRNS) from DAE-BRNS, India. We would like to extend our gratitude to all the reviewers for their valuable suggestions.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Canadian Institute for Advanced Research, Tech. Rep., 2009.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, 2015.

[4] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.

[5] K. R. Mopuri, P. K. Uppala, and R. V. Babu, "Ask, acquire, and attack: Data-free uap generation using class impressions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[6] G. K. Nayak, K. R. Mopuri, V. Shaj, V. B. Radhakrishnan, and A. Chakraborty, "Zero-shot knowledge distillation in deep networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[7] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *International Conference on Learning Representations (ICLR) Workshops*, 2014.

[8] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *International Conference on Learning Representations (ICLR) workshops*, 2015.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[10] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *International Conference on Knowledge discovery and Data mining*. ACM, 2006.

[11] R. G. Lopes, S. Fenu, and T. Starner, "Data-free knowledge distillation for deep neural networks," in *LLD Workshop at Neural Information Processing Systems (NIPS)*, 2017.

[12] H. Chen, Y. Wang, C. Xu, Z. Yang, C. Liu, B. Shi, C. Xu, C. Xu, and Q. Tian, "Data-free learning of student networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[13] P. Micaelli and A. J. Storkey, "Zero-shot knowledge transfer via adversarial belief matching," in *Advances in Neural Information Processing Systems*, 2019, pp. 9551–9561.

[14] S. Addepalli, G. K. Nayak, A. Chakraborty, and R. V. Babu, "DeGAN : Data-Enriching gan for retrieving representative samples from a trained classifier," in *Proceedings of the AAAI*, 2020.

[15] J. Yoo, M. Cho, T. Kim, and U. Kang, "Knowledge extraction with no observable data," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[16] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, "Dreaming to distill: Data-free knowledge transfer via deepinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[17] M. Haroush, I. Hubara, E. Hoffer, and D. Soudry, "The knowledge within: Methods for data-free model compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[18] X. Shoukai, L. Haokun, Z. Bohan, L. Jing, C. Jiezhang, L. Chuangrun, and T. Mingkui, "Generative low-bitwidth data free quantization," in *The European Conference on Computer Vision*, 2020.

[19] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[20] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 233–248.

[21] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.

[22] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[23] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, "Class-incremental learning via deep model consolidation," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 1131–1140.

[24] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[25] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[26] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[27] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, "A kernel method for the two-sample-problem," *Advances in neural information processing systems*, vol. 19, pp. 513–520, 2006.

[28] S. Lee, D. Kim, N. Kim, and S.-G. Jeong, "Drop to adapt: Learning discriminative features for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 91–100.

[29] J. N. Kundu, N. Venkat, R. V. Babu *et al.*, "Universal source-free domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4544–4553.

[30] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6028–6039.

[31] V. K. Kurmi, V. K. Subramanian, and V. P. Namboodiri, "Domain impression: A source data free domain adaptation method," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 615–625.

[32] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[33] K. R. Mopuri, U. Garg, and R. V. Babu, "Fast feature fool: A data independent approach to universal adversarial perturbations," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[34] K. R. Mopuri, A. Ganeshan, and R. V. Babu, "Generalizable data-free objective for crafting universal adversarial perturbations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2452–2465, 2018.

[35] K. R. Mopuri, P. Krishna, and R. V. Babu, "Ask, acquire, and attack: Data-free uap generation using class impressions," in *European Conference on Computer Vision (ECCV)*, 2018.

[36] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015. [Online]. Available: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[37] N. Balakrishnan and V. B. Nevzorov, *A primer on statistical distributions*. John Wiley & Sons, 2004.

[38] J. Lin, "On the dirichlet distribution," Master's thesis, Department of Mathematics and Statistics, Queen's University, Kingston, Ontario, Canada, 2016.

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

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015.

[42] T. Dao, A. Gu, A. J. Ratner, V. Smith, C. De Sa, and C. Ré, "A kernel theory of modern data augmentation," *arXiv preprint arXiv:1803.06084*, 2018.

[43] A. Kimura, Z. Ghahramani, K. Takeuchi, T. Iwata, and N. Ueda, "Few-shot learning of neural networks from scratch by pseudo example optimization," in *British Machine Vision Conference*, 2018.

[44] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017. [Online]. Available: https://distill.pub/2017/feature-visualization

[45] A. Mordvintsev, M. Tyka, and C. Olah, "Google deep dream," 2015. [Online]. Available: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[46] A. Madry, A. Makelov, L. Schmidt, T. Dimitris, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

[47] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[48] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[49] K. You, M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Universal domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2720–2729.

[50] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[51] S. Maji and J. Malik, "Fast and accurate digit classification," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-159, Nov 2009. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-159.html

[52] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, 2013.

[53] J. H. Eric Tzeng, "Adversarial Discriminative Domain Adaptation," Oct. 2017. [Online]. Available: https://github.com/erictzeng/adda

[54] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[55] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4068–4076.

[56] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in neural information processing systems*, 2016.

[57] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019.

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

[59] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, 2016, pp. 2234–2242.

Supplementary for "Mining *Data Impressions* from Deep Models as Substitute for the Unavailable Training Data"

APPENDIX A
CLASS IMPRESSIONS - A SPECIAL CASE OF DATA IMPRESSIONS

Class impressions by Mopuri *et al.* [35] are generated via logit maximization for the target class. On other hand, our proposed method synthesizes data impressions for a target category by matching the temperature raised softmax predictions of the Teacher network with the softmax sampled from Dirichlet distribution. Hence, our impressions are optimized using cross entropy between these softmax vectors. A special case in this context is when the target softmax output is a one hot vector. We aim to show that the synthesis of class impressions through logit maximization is same as minimizing the cross entropy loss with target softmax output as one hot vector. Therefore this shows that the data impressions are generic where the target vectors can have high or low entropy by suitably adjusting the β values whereas class impressions are a special case with target vectors of low entropy.

Notation:
T : Teacher network
x : Random Input
$T(x)$: pre-softmax values of the trained Teacher network for input x
c : Target category
$T(x)^c$: Logit value for a target category 'c'
K : Dimension of the output probability vector
CE : cross entropy
β : scaling factor
CS^c : c^{th} row of class similarity matrix CS
$\boldsymbol{\alpha}^c$: Concentration parameter for category 'c'
$I(c) = (0, \ldots, 0, 1, 0, \ldots, 0)$ is the one-hot vector with 1 for class 'c' and 0 elsewhere
CI : class impression
DI : data impression
Dir : dirichlet function

Proof:
Synthesis of CI for a category $c \in 1 \ldots K$

$\implies \max_x T(x)^c$

$\implies \max_x softmax(T(x))^c$
[$\because$ softmax is monotonic function]

$\implies \max_x 1 \cdot \log(softmax(T(x))^c)$

$\implies \min_x CE(softmax(T(x), I(c))$

$\approx \min_x CE(softmax(T(x), Dir(K, \beta \cdot \boldsymbol{\alpha}^c)$ such that $\beta \ll 1$

$\implies \min_x CE(softmax(T(x), Dir(K, \beta \cdot CS^c)$
such that $\beta \ll 1$

$\implies$ Synthesis of DI for the category c with $\beta \ll 1$

Hence, CI is a special case of DI.

APPENDIX B
COMPARISON: OUR GENERATED CLASS SIMILARITY MATRIX V/S SIMILARITY MATRIX COMPUTED USING REAL UNSEEN DATASET

We compute the class similarity matrix using two real unseen dataset. More specifically, we perform experiments with test data of cifar-10 [2] and arbitrary data sharing the same category i.e. SVHN [50]. The data is first passed to the teacher model and the features are obtained from the pre-softmax layer. Then, we perform L2 normalization on the features. We use the labels from the teacher's prediction. The features from a particular class are grouped together. Then we take the mean of the features that belong to a particular class and thus, we get the mean representative normalized feature for each class.

We obtain a class similarity matrix C where the entry in the i^{th} row and j^{th} column denoted by C_{ij} is the similarity score computed as:

$$C_{ij} = \boldsymbol{mf}_i^T \boldsymbol{mf}_j \tag{1}$$

where $\boldsymbol{mf}_i$ and $\boldsymbol{mf}_j$ are the mean of normalized features for class i and class j respectively.

Finally, class similarity matrix C is normalized through min-max normalization. This class similarity matrix is compared with our generated class similarity matrix obtained using the last layer weights of the teacher network in absence of training data as mentioned in section 3.1 (equation 1) in the main draft. The comparison is done via calculating the Pearson and Spearman correlations between them. We perform the class similarity experiments on Alexnet teacher trained on cifar-10 and the results are presented below:

TABLE 1
Correlation analysis on the class similarity matrix computed by unseen test samples of cifar-10 with respect to our generated similarity matrix.

Correlation Name	Correlation Value	p-value
Pearson	0.9490	5.8798 e-51
Spearman	0.6211	5.4070 e-12

TABLE 2
Correlation analysis on the class similarity matrix computed by unseen SVHN samples with respect to our generated similarity matrix. The teacher network's predictions are used as labels for the test samples.

Correlation Name	Correlation Value	p-value
Pearson	0.7724	4.8992 e-21
Spearman	0.5577	1.6640 e-09

As the pvalue is less than 0.05, thus the class similarity computed using unseen data (cifar-10/svhn) and our proposed approach is strong positively correlated on Pearson correlation and moderate positively correlated on Spearman correlation.