Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра математики факультету інформатики

Адаптація контексту у задачах класифікації зображень

Курсова робота

за спеціальністю „Прикладна математика ” 113

Керівник курсової роботи
к.ф.-м.н., ст.в. Швай Н.О.

_____
(підпис)

“____” _____ 2021 р.

Виконав
студент 1 курсу МП
факультету інформатики
Крошин О. А.

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри математики,
проф., д.ф.-м.н.
_____Б. В. Олійник
(підпис)
„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту Крошину О. А. факультету інформатики 1-го курсу МП
ТЕМА Адаптація контексту у задачах класифікації зображень
Зміст курсової роботи:

1. Індивідуальне завдання
2. Календарний план
3. Анотація
4. Вступ
5. РОЗДІЛ 1: Deep Visual domain adaptation, an overview
6. РОЗДІЛ 2: Self-ensembling for visual domain adaptation
7. Висновки
8. Список використаних джерел

Дата видачі „___” _____ 2020 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання теми курсової роботи.	10.10.2020	
2.	Огляд задачі	12.12.2020	
3.	Збір даних для проведення досліджень	01.03.2021	
4.	Написання першого розділу	01.04.2021	
5.	Проведення досліджень	15.04.2021	
6.	Написання другого розділу	01.05.2021	
8.	Корегування роботи згідно із зауваженнями керівника	10.05.2021	

Студенту Крошину О.А.

Керівник Швай Н.О.

“_____”_____

Table of contents

Abstract

Rapid developments in the Deep Learning domain in recent years let researchers and practitioners shift their focus from training machine learning models itself to transferring the already-learnt knowledge and applying it in different applications. This paper discusses Domain Adaptation, a subdomain of transfer learning, primarily aimed at applying knowledge from a given source domain to an unknow target one. It discusses various Domain Adaptation settings under the context of Computer Vision, introduces self-ensembling Domain Adaptation methods for semi-supervised learning and illustrates its capabilities with proper experiments.

Experiments were implemented with Python 3.6 using libraries pytorch, numpy, pandas, opencv, matplotlib, torch-salad, etc.

Introduction

Recent developments in machine learning world have greatly improved usability of created models in everyday tasks. However, due to the costs, retraining and annotating unseen data for a particular solution remains a huge obstacle for researchers and practitioners.

The aim of this paper is to review Deep Domain Adaptation, a set of techniques, aimed at adopting models to new tasks and datasets without fully retraining them from scratch, give an overview for a variety of different Domain Adaptation tasks, as well as to provide intuition behind Domain Adaptation methods to be used under particular settings.

First part of the paper is focused on an overview of various Domain Adaptation settings, specifically in context of Computer Vision subdomain and from a narrower perspective of an Image Classification problem. Second part introduces a variety of self-ensembling methods for semi-supervised visual domain adaptation in a setting of image classification. The final, third part, describes datasets and certain tricks necessary for a successful training before interpreting results of different self-ensembling DA methods using SVHN and MNIST datasets.

1. Deep Visual Domain adaptation, an overview

1.1. Domain adaptation in a context of transfer learning.

The problem of unlabeled data is one of the scourges of the machine learning world. Collecting, transforming and annotating new data of high-quality is one of the toughest obstacles between a data science team and a production-ready machine learning solution. Retraining a model from scratch is another regular challenge: modern state-of-the-art solutions in challenging domains are built on complex neural-based architectures with lots of layers. Another aspect of their success is pre-training on datasets containing millions of training pairs. In example, latest solutions used in an image captioning domain are pre-trained on millions of images. Adopting such models for new distributions of the unseen data using simple fine-tuning may be also a challenge, as it will require labels and lots of computational power in the first place.

Using knowledge learnt by machine learning model while training for a particular task in order to apply it to another machine learning task is a sphere of transfer learning (TL) research problem. Given a domain $\boldsymbol{D}$, which consists of feature space $\boldsymbol{F}$ and marginal probability distribution $\boldsymbol{P(X)}, \boldsymbol{X} \in \boldsymbol{F}$. For a given $D = \{F, P(X)\}$, the learning task is defined as $T = \{Y, f: F \rightarrow Y\}$, where Y is a label space, and f is an objective function which may be seen as $P(Y|X)$. The machine learning model learns how to predict $y_i \in Y$ given training pairs $\{x_i, y_i\}$. With source domain and task D_s, T_s, the aim of transfer learning is to enhance performance for the objective function f_t for a target task T_t in a domain D_t.

According to Pan et al [6], transfer learning approaches may be categorized into three main categories: transductive, inductive, and unsupervised.

In a situation when $P_S(X) = P_D(X)$ and $T_s \neq T_t$, transfer learning tasks are defined as inductive. Alternatively, given that $P_S(X) \neq P_D(X)$ and $T_s = T_t$, transfer learning approaches are categorized as transductive. Finally, in situations where $P_s(X) \neq P_D(X)$ and $T_s \neq T_t$, unsupervised transfer learning techniques are applied.

The field of Domain Adaptation (DA) is associated with transductive transfer learning, and focuses on adapting machine learning model for a new source distribution. The examples of Domain Adaptation may be applying diagnostic systems to identify new diseases based on experience of identifying similar ones, adopting sales forecasting model to a new region, etc.

Deep Domain Adaptation is a set of methods that use deep networks to improve DA performance.

1.2. Classification of Domain Adaptation settings

Depending on whether the source and target domains are directly related, adapting to an intermediate (transitive) domain may be necessary in a domain adaptation process. Based on the necessity of transitive domain adaptation step, DA may be classified as one-step or multi-step.

Based on whether source and target feature spaces F_s, F_t are identical or not, DA ca be classified as homogeneous ($F_s = F_t$, $d_s = d_t$, where d represents feature space dimensionality), and heterogeneous ($F_s \neq F_t$).

Finally, DA may be classified as supervised, semi-supervised or unsupervised based on the nature of target domain training data:

- In the supervised DA, some labeled target data D_{tl} is present. However, D_{tl} is insufficient to reach great level of performance.
- In the semi-supervised DA, D_{tl} is present along with target domain unlabeled data D_{tu}, which is sufficient for learning D_t structure information.
- In the unsupervised DA, only D_{tu} is available.

DA classification diagram is given on a figure 1

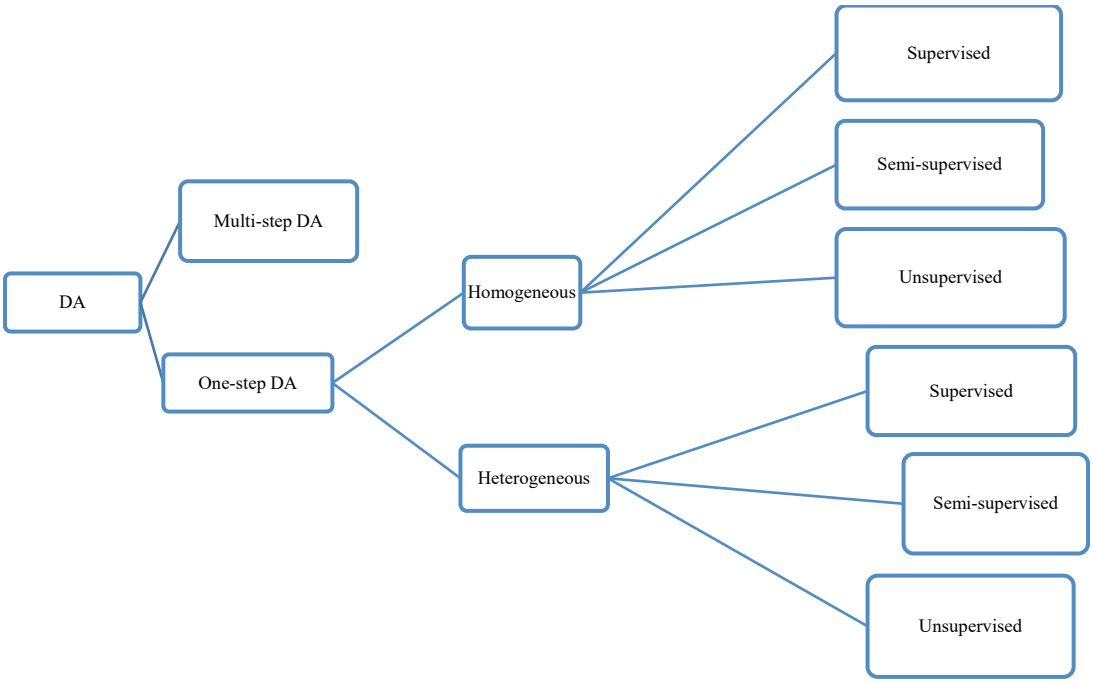

Figure 1 - Domain Adaptation Classification diagram

It may be seen that with the increase of K grows the computational resource to reach the stable state of the system. Kauffman demonstrated that the system becomes more chaotic. Note that in this paper we assume that T=2 unless other value of T is specified.

1.3 Deep Domain Adaptation: Approaches

1.3.1 One-Step Domain Adaptation

The basic idea behind one-step deep DA lies in utilizing deep networks in order to learn "good" and domain invariant feature space using available source and target domain data.

Deep approaches in one-step DA can be categorized into three subgroups:

- Discrepancy-based approaches focus on fine-tuning the network in order to reduce the distributional shift between the domains.
- Reconstruction-based approaches use techniques of data reconstruction in order to reduce variance between feature as much as possible
- Adversarial-based approaches

Each group of the approaches has its own group of techniques for performing an assigned task.

1.3.1.1 Discrepancy-based DA

Discrepancy-based DA consists of four major techniques for performing fine-tuning:

1. Architecture criterion – adapts neural networks to learning wider sets of features, which are less domain-specific and may potentially be used among multiple domains. Such techniques include adaptive batch normalization, domain-guided dropout, etc.
2. Class criterion – transfers interdomain knowledge via relying on data labels. When D_{tl} are available, metric learning under soft

labels is almost always efficient. Otherwise, one may use methods of assigning pseudo labels, or using certain features as labels.

3. Geometric criterion – is used under the assumption that relationship of source and target domain may be explained under some geometrical properties. Can narrow the gap between source and target domains via exploiting some of its geometric properties.
4. Statistic criterion – uses statistical methods to adapt source and target domains to one another via minimizing the calculated metric. Among possible methods are correlation alignment (CORAL), KL-divergence, and other criteria that measure similarity between two probability distributions.

In heterogeneous DA scenarios, images in different domains have to be resized into same dimensions in order for Class or Statistic Criterions to be utilized.

1.3.1.2 Reconstruction-based DA

Given its name, Reconstruction-based DA is based on an assumption of the utility of data reconstruction for source and/or target samples. Reconstruction-based DA approaches can be classified as follows:

1. Adversarial reconstruction – minimizes the reconstruction error – a difference between original and reconstructed images in each domain. Utilizes a discriminator like a cycle GAN.
2. Encoder-Decoder Reconstruction: encoder-decoder reconstruction methods utilize the encoder for representation learning, and the decoder for data reconstruction.

1.3.1.3 Adversarial-based DA

In Adversarial-based DA, a domain discriminator classifies if a point belongs to a D_s or D_t, and an adversarial objective is designed to minimize the gap between D_s and D_t distributions. Its approaches can be categorized into two types of techniques:

1. Generative models – generative adversarial networks (GANs) may be used to generate samples that similar to ones from target distribution, and yet store the labels of the source distribution at the same time.
2. Non-Generative models – given labels from D_{sl}, an image feature extractor is trained a discriminative representation to create domain-invariant features, which are then used to map the target data to the source distribution.

1.3.1.4 One-step DA approach comparison table

		Supervised DA	Unsupervised DA
Adversarial-based	Generative Models		+
	Non-generative models		+
Reconstruction-based	Adversarial Models		+
	Encoder-Decoder Models		+
Discrepancy-based	Architecture Criterion	+	+
	Class Criterion	+	
	Geometric Criterion	+	
	Statistic Criterion		+

Table 1 – DA approaches comparison table

1.3.2 Multi-Step Domain Adaptation

When working with multi-step DA, the initial step is to identify intermediate domains that are "closer" to D_s and D_t than source and target domains are with each other. Naturally, one-step DA will be applied on each step between source, target and transitive domains. The most challenging part of multi-step DA lies in identifying the most useful intermediate domains possible and applying them correctly.

Multi-step DA can be classified into three categories:

- Hand-crafted: intermediate domains are selected by the users.
- Instance-based: intermediate domains are enriched with additional data from auxiliary datasets.
- Representation-based: firstly, pretrained network is freezed, and then its output is used as an input to the new deep network.

1.4 Deep domain adaptation and image classification

Lots of the techniques described above may be used in a wide variety of computer vision applications, among those in object detection, semantic segmentation, style transfer, etc. However, image classification is the basic problem for which most of the deep DA approaches were proposed.

One may define an image classification problem as a task of selecting the most probable label(s) from a pre-defined set of labels given an image as an input.

Image classification problem provides lots of benchmarks for different tasks. In example, for deep DA different techniques are compared by analyzing certain metrics (mainly, accuracy) using different datasets as D_s and D_t. Among those benchmark datasets are so-called digits dataset, mainly MNIST, USPS and SVHN, which will be described in more detail in part 3 of this work. Another important benchmark is the Office-31 dataset, which consists of images from three different domains: Amazon (A), DSLR (D), and Webcam (W). There are 31 object classes. The A domain consists of images of office objects taken from online stores, The D domain contains photos of high-quality office objects, ant the W domain consists of low-quality images with lots of noise and artifacts.

2. Self-ensembling for visual domain adaptation

2.1 Problem description

Among different practical settings available for visual deep DA, one of the most common scenarios is when a source domain dataset is fully labeled, and a target dataset consist of a small labeled part and lots of unlabeled examples. This setting is called semi-supervised domain adaptation. In this section we will review some recent state-of-the-art solutions in the area and discuss them in detail.

2.2 Self-ensembling for semi-supervised learning

The problem behind semi-ensembling learning techniques is closely related to the connection between vision and perception. When a human sees an object under different conditions, it typically still recognizes object class correctly. Similar to a human, a machine learning algorithm should also be able to correctly identify objects after some perturbations and changes. Methods that are focused on enhancing model's ability to detect objects correctly under different conditions are known as consistency-enforcing approaches,

Application of regularization techniques such as Dropout or augmentations is a simple way to improve stability of model predictions. In a semi-supervised setting model stability may be improved via evaluating labeled examples with and without regularization. The approach, originally proposed as Γ-model, used the same model as two parts: a teacher model would generate targets, and a student model would use generated targets for training. An immediate problem would be the quality of targets, as generating too inconsistent augmentations would change training examples too much and only worsen classification

performance. Other than carefully selecting regularization steps to be used, one could improve teacher's network to generate better examples and tune student network to learn faster. This class of semi-supervised learning methods is known as self-ensembling.

Below we will describe state-of-the-art methods that use the aforementioned approach. We will start with analyzing the Π-model and Temporal ensembling, proposed by Laine & Aila [5], and examine possible additional improvements under the Mean Teacher model, introduced by French et al. [4].

General idea behind all three models lies in getting separate class probability distributions z_i and $\tilde{z}_i$ from teacher and student networks given a single input image, and then minimizing a "distance" between two distributions given a label, which matches target distribution to a source domain.

2.2.1 Self-ensembling Loss function

A loss function used in proposed methods is called consistency loss. It consists of supervised an unsupervised part. It is worth mentioning that the original problem focuses on image classification task with partially labeled data. In context of DA, labeled subsets from D_s and D_t may be used and evaluated separately in order to preserve separate normalization statistics for each domain.

In a traditional classification setting, labels from source are evaluated on cross-entropy, and all source and target labels are evaluated on consistency loss, which is calculated using mean square difference between z_i and $\tilde{z}_i$. Supervised and unsupervised loss are then combined via applying weighting function $w(t)$, dependent on time.

Let us define cross-entropy and consistency losses for a classification problem given minibatch B, a total of C classes and a parameter λ which controls importance of consistency part:

$$l_{CE}(z_i, y_i) = -\frac{1}{|B|}\sum_i \log\ \mathrm{z_i}[y_i]$$

$$l_{cons}(z_i, \tilde{z}_i) = \frac{1}{C|B|}\sum_i \left|\left|z - \tilde{z}_i\right|\right|^2$$

We may define total loss as:

$$L(w_f) = l_{CE}(z_i, y_i) + \alpha w(t) l_{cons}(z_i, \tilde{z}_i)$$

Given a number of ramp-up steps n_e, a weighting function $w(t)$ is defined as: $w(t) = r * w_{max}$, where r is a Gaussian ramp-up scaling factor and w_{max} is a number of labeled training inputs divided by total number of training inputs:

$$w_{max} = \frac{|D_{sl}|}{|D_{sl}|+|D_t|},\ r = \begin{cases} \exp(-5\mathrm{p}^2)\,,\ \mathrm{p} = 1 - \max\left(0, \frac{\mathrm{epoch}}{n_e}\right), \text{if epoch} < \mathrm{n_e} \\ 1, otherwise \end{cases}$$

Note that in practice for different datasets and models w_{max} parameter may be chosen arbitrarily.

2.2.1 Π-model

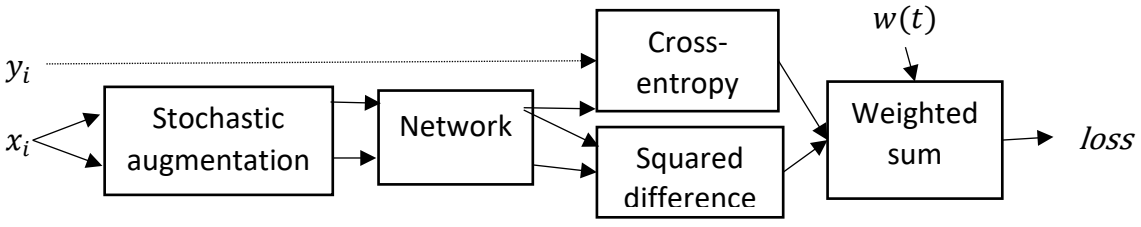

Figure 2 - pi-model training structure

The first of the approaches, the Π-model uses a single network both for a teacher and student. Each training step consists of applying two different random perturbation sets to an input image x_i in order to get class distribution vectors z_i and $\tilde{z}_i$.

A proposed consistency loss utilizes $\lambda = 1$:

$$L(w_f) = l_{CE}(z_i, y_i) + w(t) l_{cons}(z_i, \tilde{z}_i)$$

Structure of Π-model training pass is summarized on Figure x:

2.2.2 Temporal ensembling model

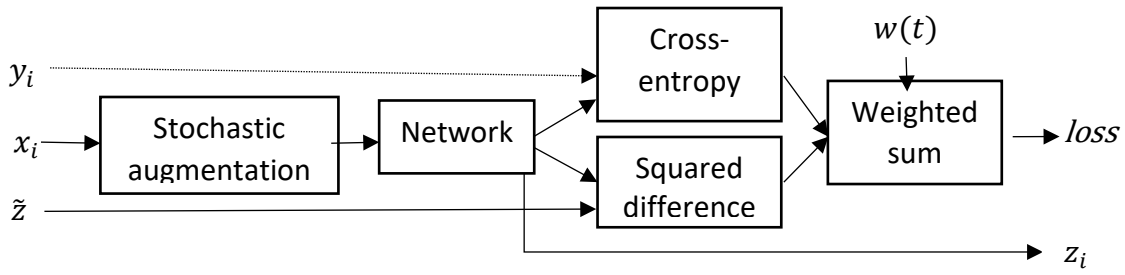

Figure 3 - Temporal ensembling model training scheme

In order to store information about previous training results, the evaluation of z_i and $\tilde{z}_i$ can be split into two steps. Given that input images are noised after every pass through a model input, we can calculate z_i only once per epoch (input will be noised anyway), and then mix z_i with previous evaluation output.

Using temporal ensembling in favor of Π-model may be beneficial in several ways:

1. Training targets will be less noisy, which will benefit inference performance on cleaner inputs
2. Calculating z_i only once will speed up training

3. Storing z_i at different timesteps enables collecting different statistics and possibly using it in future enhancements (i.e., select only steps with lower variance)

As of the downsides of the model, learnt information is absorbed by a student network slowly (only once per epoch), and utilizing predicted distributions in case of large datasets with lots of training epochs is questionable.

Temporal ensembling training pass is summarized on Figure 3.

2.2.3 Mean Teacher model

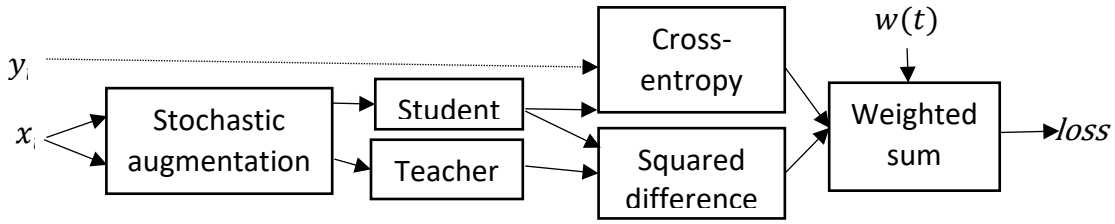

Figure 4 – Mean Teacher model training structure

Limitations of temporal ensembling model made Tarvainen & Valpola [3] propose an approach of averaging model weights instead of class prediction distribution. As the approach focuses on presenting a teacher model as an average of consecutive student models, an approach was called a Mean Teacher method.

For a student model with weights θ and a teacher model with weights θ' and a given smoothing coefficient α,we update teacher model weights at step t as an exponential moving average of student weights:

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$$

Note that in this approach teacher weights θ' are treated as a constant in terms of optimization.

2.3. Adopting Self-ensembling to Domain Adaptation

2.3.1. Mean Teacher model

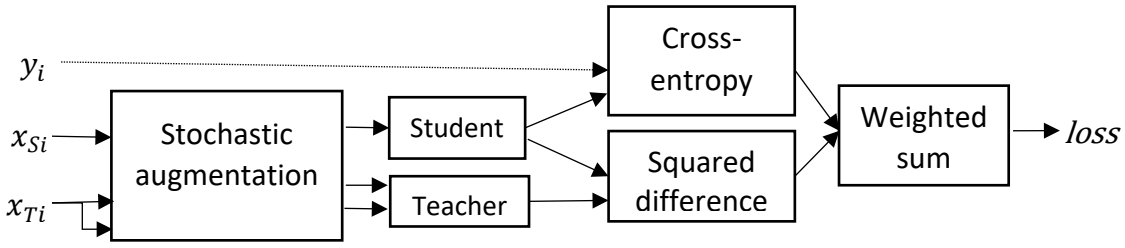

Figure 5 – Mean Teacher model adopted to DA

In order to use Mean Teacher model in DA setting, it has to be adapted in a following way:

- Labeled samples from D_S are trained via cross-entropy
- Samples from D_T are trained in an unsupervised manner via consistency loss, mentioned at 2.2.1

2.3.1. Confidence thresholding

Experiments, conducted by French et. al. [6], demonstrated that using confidence thresholding instead of the ramp-up Gaussian weighting factor in the unsupervised loss may improve training stability under different scenarios.

Confidence thresholding technique uses a searched f t to mask a sample x_i to zero if a condition is match:

Given an unlabeled example x_{T_i} and the corresponding predictions vector $\tilde{z}_{T_i}$, we select a predicted probability $\mathrm{f_{T_i}} = \max_{j \in C} \tilde{z}_{T_i}$ for a selected class j of the given example. If $f_{T_i} < t, x_i$ is set to zero.

2.3.2. Class balance loss

In case of highly imbalanced datasets, Mean Teacher model may fall into a local minima, separate source and target datasets, and disproportionately favor single class one of the datasets.

This problem can be tackled by using a class balance loss term, which penalizes the network for making predictions with highly-imbalanced distribution. For each mini-batch of x_{T_i}, we calculate mean per-class probability vector m_i. Then, cross-entropy loss $l_{CE}(m_i, z_i)$, where z_i is a uniform distribution of class probabilities.

Final loss is calculated as $L = kl_{CE}(m_i, z_i) + L'$, where k is calculated as a mean of the confidence threshold mask.

3. Practical research

3.1. Used datasets

3.1.1. MNIST

Figure 6 - samples from MNIST dataset

Introduced by National Institute of Standards and Technology of the USA, full MNIST is a classic introductory image classification dataset that consists of 70000 grayscale hand-written digits, each image of size 28x28 pixels. During experiments, we use training set of 60000 images. In order to use grayscale 28x28 images of MNIST along with colored data from SVHN, we had to pad images to 32x32 and convert grayscale to RGB.

Figure 7 - padded MNIST samples

3.1.2. SVHN

Figure 8 - samples from SVHN dataset

Google's Street View House Number Dataset consists of 630414 RGB images of digits, each being a cropped part of house number plates of size 32x32. SVHN consists of train set of 73257, test set of 26032, and an extra set with 530000 additional images that are less difficult to identify. We use SVHN train and test subsets in our experiments, and didn't add any transformations during pre-training stage.

3.2. Experiments

Experiments were conducted on SVHN and MNIST datasets, where SVHN was used as a source domain D_s, and MNIST was used as a target domain D_t. The aim of the experiments was to compare performance of proposed self-ensembling models, mainly Mean Teacher model, Π-model and Temporal Ensembling model. As a baseline, we've implemented a two-layer classifier based on MobileNet v2 Backbone, trained it on SVHN-train dataset and evaluated on MNIST-test dataset, both with and without augmentations. As one would expect, a classifier perform badly, reaching accuracy of 0.41 on MNIST test set with augmentations in place, and accuracy of 0.34 without such augmentations.

3.2.1. General neural architecture

For experiments between MNIST and SVHN we used the following neural network architecture:

Description	Shape
Input image	32x32x3
Conv 3x3x128 with padding=1, batch-norm	32x32x128
Conv 3x3x128 with padding=1, batch-norm	32x32x128
Conv 3x3x128 with padding=1, batch-norm	32x32x128
Max-pooling, 2x2	16x16x128
Dropout, p=0.5	16x16x128
Conv 3x3x256 with padding=1, batch-norm	16x16x256
Conv 3x3x256 with padding=1, batch-norm	16x16x256
Conv 3x3x256 with padding=1, batch-norm	16x16x256
Max-pooling, 2x2	8x8x256
Dropout, p=0.5	8x8x256
Conv 3x3x512 with padding=0, batch-norm	6x6x512
Conv 1x1x256, batch-norm	6x6x256
Conv 1x1x128, batch-norm	6x6x128
Global pooling layer	1x1x128
Dense layer, 10 units after softmax	10

Table 2 - NN architecture used in experiments

3.2.1. Data Augmentation

In order to increase our chances to achie state-of-the-art results on the experimental datasets, we used the affine scheme that adds random transformations according to a matrix:

$$\begin{bmatrix} 1 + N(0,0.1) & N(0,0.1) \\ N(0,0.1) & 1 + N(0,0.1) \end{bmatrix}$$

It is worth mentioning that impact of data augmentation techniques should not be underestimated, and to illustrate that statement we will run separate experiments, ones using aforementioned affine scheme, and others via applying simple Gaussian noise with $\delta = 0.1$.

3.2.1. Experiments settings

Our main experiment would be in comparing results of self-ensembling models on the aforementioned datasets. In particular, we test performance of Π-model, along with Mean Teacher model with different smoothing coefficient α. We will also show how advanced augmentation techniques enhance model performance. Student accuracy on source and target domains will be used as a main performance metric, along with the combined self-ensembling loss.

All the models were trained 100 epochs with a fixed initial random seed. However, we propose analyzing the results after 25 epochs, as under 100 epochs accuracy for different parameters is too close to 1 and loss is almost 0, which makes explaining results a difficult task.

3.2.2 Results comparison table

	Augmentation Affine Scheme	Studetnt accuracy,Source	Student accuracy, Target	Teacher accuracy, target	Combined Loss	Note
Π-model	+	0.885	0.89	0.985	0.0006*	
Mean Teacher model, $\alpha = 0.95$	-	0.8928	0.8812	0.965	0.0912	
Mean Teacher model, $\alpha = 0.95$	+	0.9713	0.9815	0.9817	0.0266	
Mean Teacher model, $\alpha = 0.99$	-	0.9215	0.9324	0.9724	0.0334	
Mean Teacher model, $\alpha = 0.99$	+	0.9817	0.9713	0.9892	0.0026	
Mean Teacher model, $\alpha = 0.999$	+	0.9349	0.9192	0.9765	0.0733	

Table 3 – comparison of experiment results

3.3. Results

Obtained results clearly demonstrate an advantage of using enhanced image augmentation schemes. Alongside, it may be seen that without using the

weights of the Teacher model, the Π-model performs poorer than a Mean Teacher model, despite showing good results overall.

As was shown in [4], $\alpha = 0.99$ is considered the most suitable for a majority of domain adaptation tasks. However, in the implementation of their paper, the authors suggest using $\alpha = 0.999$ for smaller datasets. Unfortunately, SVHN -> MNIST transfer performs weaker using increased α parameter.

Summary

This work covers a topic of deep domain adaptation, reviews its various settings depending on presence of data, homogeneity of source and target domains and approaches available for a particular domain adaptation task. The work analyzes self-ensembling techniques for semi-supervised learning for an Image Classification problem and compares different self-ensembling models along with their reported performance.

During experimental phase, we compare performance of Mean Teacher model and Π-model in an SVHN -> MNIST domain adaptation setting. Obtained results demonstrate utility of the aforementioned self-ensembling solutions, as well as the importance of proper augmentations and correct parameter selection for a successful training process.

To sum up, we can confirm the validity of using self-ensembling methods for semi-supervised deep visual Domain Adaptation.

References

1. Wang, M. - Deep Visual Domain Adaptation: A Survey / Wang, M., Deng. W. - *Neurocomputing*, 2018, 312: 135-153, doi:10.1016/j.neucom.2018.05.083

2. Géron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow; – O'Reilly [2017] – 542c.

3. Tarvainen,A. Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, 2017.

4. French, M. Mackiewicz, and M. Fisher. Self-ensembling for visual domain adaptation. *International Conference on Learning Representations*, 2018.

5. Laine, S., Alia, T. Temporal Ensembling for Semi-Supervised Learning – *International Conference on Learning Representations 2017*.

6. Pan, S., Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.