Evolutionary Strategies for Domain Adaptation
CE888 Data Science and Decision Making

Tanya Ramirez Franco - 1702557

***Abstract*— This project will propose a transfer learning approach directly inspired by the theory of domain adaptation where the data at training and testing time comes from distributions that are similar to each other but have relevant differences. The aim of the project is to discuss a system for classification that learns features that fail to discriminate between source and target distribution while being able to learn correctly the mapping between the distribution of the source and target data and their labels. The aforementioned objective will be tested by creating a neural network that will take as input the data provided and outputs a set of features that will be used to train a random forest. Then, an evolutionary process will start using SNES to adapt the weight in the neural network. The experimentation was applied in two different data sets (MNIST-M and OFFICE-31) and the results were compared with experiments from other authors. The results obtained left space for improvement and further analysis given that in certain cases, the overall evaluation did not achieve the expectations. Nevertheless, the Domain Adaptation in general requires further research since labeled data in the real world is scarce, the possibility of using data with different domains appears as a reliable solution to be engaged.**

***Index Terms*— Domain Adaptation, Transfer Learning, Neural Networks, Classification, Unsupervised Learning, Separable Natural Evolution Strategies**

I. INTRODUCTION

Domain adaptation goal is to create learning algorithms that can be transferred from one specific domain to other different domain but which has some similarities to the first one. In other words, this means that the ”learning” comes from a source data that has a different distribution but related to the target data distribution. This type of problems arises in different situations as for example in object recognition where the training and evaluation are performed on the same image distribution, but when the model is actually applied on the real world, there is a change in the visual domain caused by unknown factors like scene, intra-category variation, pose, view angle, camera characteristics, etc. Furthermore, studies have shown that there is a significant deterioration in the performance of image classifiers because of the shift from pose changes[7] or more generally, training datasets that are biased by the way in which they are collected[16].

As it is defined in [8], domain adaptation happens when the task is to learn a discriminator classifier or other predictor in the presence of a shift between training and test distributions. In the case where the source data comes from different domains, it is defined as a multidomain adaptation problem.

Fig. 1. Example of application of Domain Adaptation - Object recognition[13]

There are many different situations where domain adaptation is a useful tool. For example, in the context of sentiment analysis it can be the case where the source data is the labeled data reviews of one type of product and the goal is to classify reviews of other products[9]. Another example can be seen when the task is to re-identify a person (associate people seen from different camera views): There are two sets of images from different cameras such that each person is represented in a probe data set as an image in a gallery data set and the task is such that for each picture of a person from the probe set, find an image of the same person in the gallery set. Domain adaptation can represent a feasible mechanism to improve deep re-identification descriptors[9].

The principal motivation of further study in domain adaptation is that high quality data sets (in scale and caliber) are costly and sometimes acquiring more data represents an impossible task. For example, in order to acquire 3D pose labels or real-world images for robotics can be an extremely expense task. One approach that has been take recently is to generate a high number of synthetic data and with domain adaptation, transform it as it had been acquired from the real world[5]. Generally speaking, the cost of generating labeled data is a tremendous deterrent for apply machine learning methods specially recently with the progress of the use of deep neural network architectures and the necessity of huge amounts of data.

The aforementioned problems are just a few examples where domain adaptation can provide a feasible solution and contribute to the development of useful learning algorithms. The aim of this report is to create a system that learns features that fail to discriminate between source and target distributions (domain divergence) but at the same time are able to learn correctly the mapping between those distributions (source and target) and their labels. The proposed methodology will be in first place to create a neural network that takes as input the data and produces as output a set of features that will be used to train random forests. Then, using a evolutionary process (SNES) the weights of the neural network will be adapted. The scores that will be given back to the SNES will provide an intuition of how poorly a random forest fails to discriminate between the source and target domain and at the same time, will provide an intuition of how well it discriminate between the different classes. The data sets that will be used are the MNIST-M and OFFICE-31. For this last data set, three scenario were analyzed by changing the source and target domain.

The essence of this report and part of the support analysis is based on [9]. Nevertheless, the main difference is to encode the Domain Adversarial Neural Network loss function within an Evolutionary Algorithm and use Random Forest to replace the dense layers of the Neural Networks. The loss for every generation will be reviewed and the overall results of the model proposed for each data set from the proposed model will be compared with the results from [9].

II. BACKGROUND

A. Transfer Learning

In order to understand better the implementation of domain adaptation, the concept of transfer learning must be reviewed before. The objective of the machine learning algorithms is to make the most accurate prediction in future observations based on models that were trained on labeled or unlabeled data during the training phase. There is one approach called semi-supervised learning that tries to take advantage of situations were the labeled data is scarce but unlabelled data is easy and cheap to collect. But still in these cases, an important assumption is made: the distribution of the labeled and unlabeled data are the same. The following graphic shows the difference between the common approach used in machine learning and the approach used in transfer learning models.

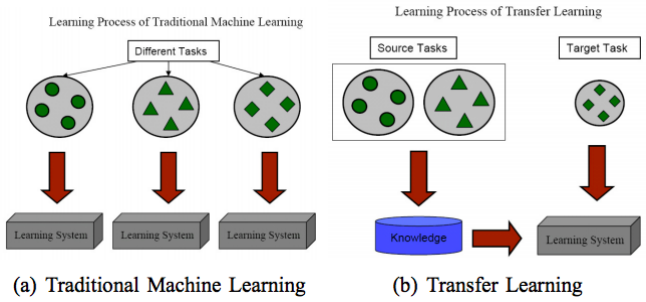

Fig. 2. Differences between Machine Learning and Transfer Learning [12]

On the other hand and as it is mentioned in [12], transfer learning "allows the domains, tasks, and distributions used in training and testing to be different". This approach is more similar as what is found in the real word and can be useful for simulations that require hardware for the training of a machine learning algorithm (for example, self-driving cars).

A most formal definition of transfer learning can be as is described in [12]. Define $\mathscr{D}_S$ as the source domain and $\mathscr{T}_S$ as the learning task of this domain. Similarly, define $\mathscr{D}_T$ as the target domain and $\mathscr{T}_T$ as its learning task. Transfer learning tries to improve the learning of the target predictive function $f_T(\cdot)$ in $\mathscr{D}_T$ by incorporating the information and knowledge in $\mathscr{D}_S$ and $\mathscr{T}_S$, where $\mathscr{D}_T \neq \mathscr{D}_S$, or $\mathscr{T}_T \neq \mathscr{T}_S$.

The domain is in the form of $\mathscr{D} = \mathscr{X}, P(X)$ and, similarly, a task is defined as $\mathscr{T} = \mathscr{Y}, P(Y|X)$. When the target and source domain are the same and their correspondent tasks are too, the learning problem is simply a machine learning problem. Nevertheless, when the domains are different, there are two different cases to explore:

1) $\mathscr{X}_S \neq \mathscr{X}_T$ meaning that the feature spaces between the domains are different; or

2) $P(X_S) \neq P(X_T), X_S \in \mathscr{X}_S, X_T \in \mathscr{X}_T$ meaning that the features spaces between the domains are the same but the marginal probability distributions between domain data are different.

The most common categorization based on the characterization and relationship between source and target domain and tasks of transfer learning is the following:

- Inductive transfer learning;
- Transductive transfer learning; and
- Unsupervised transfer learning.

For its nature, Domain Adaptation fails in the category of transductive transfer learning.

B. Transductive Transfer Learning

In comparison to machine learning and the other types of transfer learning, in transductive learning the source and target domain are different but related in some way. Furthermore, the task is the same for both domains (source and target). Transductive learning can be understood as a situation where the test data are required to be seen during the training of the model, and in this case, "the learned model cannon be reused for future data"[12]. In this way when there is new test data to predict, these data need to be classified along with the previous data.

In the case of transfer learning, transductive transfer learning requires that part of the unlabeled target data set be seen at training time so the model can obtain the marginal probability for the target data and as it was mentioned before, the tasks in this case must be the same for the source and target domain and there must be some unlabeled data available in the target domain.

C. Domain Adaptation

In order to define conceptually what is the objective of domain adaptation, consider X as the input space and $Y = [1, 2, \ldots, L]$ is defined as the set of possible labels. Furthermore, the are two different distributions over $X \times Y$. Define $\mathscr{D}_S$ as the source domain and $\mathscr{D}_T$ as the target domain.

An *unsupervised domain adaptation* learning system is then provided with a *labeled source sample* S drawn $i.i.d.$ from $\mathscr{D}_S$, and an *unlabeled target sample* T drawn $i.i.d.$ from $\mathscr{D}_S^X$ as a marginal distribution of $\mathscr{D}_S$ over X.

$$S = (x_i^s, y_i^s)_{i=1}^n \sim (\mathscr{D}_S)^n;$$
$$T = (x_i^t, y_i^t)_{i=n+1}^N \sim (\mathscr{D}_T^X)^n$$

with $N = n + n$ being the total number of samples. The problem consists in finding a classifier with a low target risk without having information about the labels of $\mathscr{D}_T$.

In other words, the task is to train a model on a source data set for which there is a labelled sample and apply this model on a target data set that was generated from a different but with similar characteristics distribution. The definition of the system can be created using different characteristics of $\mathscr{D}_T$:

- Unsupervised domain adaptation where the target domain does not have any labels; and
- Semi-supervised domain adaptation when there are fewer labels on the target domain that in the source domain.

D. Domain Divergence

Given that the objective of the system is to discriminate between source and target distributions, many approaches have bounded the target error by the sum of the source error and a notion of distance between the source and the target distributions, as it was mentioned before. Intuitively, the source risk is expected to be a good indicator of the target risk when both distributions are similar. So, there must be a notion of distance specifically for domain adaptation. As a first instance, the intention of this report is to use $\mathscr{H}$-divergence used by [4] and [3].

E. Separable Natural Evolution Strategies (SNES)

Evolutionary Strategies (ES) produce consecutive generations of samples and during each generation, a batch of samples is generated by perturbing the parents' parameter. A number of samples is selected based on their fitness values while the less fit individuals are discarded. The "winners" are then used as parents for the next generation and so on. A method that tries to include these low-fitness examples and use all available examples to generate a gradient for updating the population is Natural Evolution Strategies (NES) which adapts both a mutation matrix and the parent individual using a natural gradient based update step. In this way, every generation, a gradient towards better expected fitness is estimated using a Monte Carlo Approximation. Standard gradient methods have been shown to converge slowly[17]. In a few words, NES are a class of ES for real-valued optimization that has a search distribution and adapt the distribution parameters by following the natural gradient of expected fitness. Nevertheless, a shortcoming of the NES is its limited applicability to high dimensional problems. *Separable Natural Evolution Strategies* (SNES) comes as an alternative given that it allows for linear time updated

and makes the algorithm applicable to extremely high-dimensional problems just having as an assumption that the problem variables are mostly separable[14].

F. Random Forest

As it was mentioned before, the output of a Neural Network will be used to train a Random Forest where its performance is determined by how bad it fails to discriminate between the source and target data and how well it discriminates between the different classes. It is important to remember that in order to guarantee the correct application of the model, the features in the source and target domain with their correspondent distributions must be related to each other.

As it was mentioned before, the output of the Random Forest is the classification of the labels and a non-discrimination classification between domains. As it is known, a decision tree follows a greedy approach to recursively partition the data based on some feature value test. Each leaf node in a decision tree follows a distribution say P of class labels that is associated with a set of data partition. As it is mentioned in [15], "A path in a decision tree from the root to a leaf node contains a sequence of features chosen as split function". In this way, the Random Forest trained for this model is constantly updated from the resulting weights that are adapted to the Neural Network that outputs the set of features.

Furthermore, one of the principal advantages of the implementation of the Random Forest as classifier is that in this model, the labels across the source and target domain can be estimated juts using a single model and in this way the complexity of the domain adaptation approach is reduced proving a more approachable understanding of the model and the relationships.

In recent years, a large number of methods of domain adaptation have been proposed in different fields. For example, some models have suggested to apply techniques based on cross-domain transformations where the idea is to learn a regularizer non-linear transformation that maps from the source domain to the target domain using supervised data from both domains. This transform-based approach can be applied over novel test samples from categories seen at the training time but also can be applied to new categories which were not present at training time[13]. Other approach that has been proposed in [6] applies a domain adaptation approach by transforming the features into an augmented space. The input features from each domain are copied twice: To the domain-independent portion of the feature vector and to the portion specific to that domain. The main hurdle in this approach is that it only works for classifier that only learn over the features.

More specifically, there are some proposed methods that also incorporate the goal to match the feature distribution in the source. A parallel research proposed by [1] proposed a neural network algorithm that introduced the idea that the hidden layer should learn a representation that predicts the source example labels but is not aware of the domain of the input (no difference between coming from the source or target domain). For the purpose of this project and in order to make a comparison of the results, the research from [8] will be contrasted. At the end, the purpose of the model of the report and in [8] is similar: Create a system that discriminates for the learning tasks on the source domain and is invariant with respect to the different domains.

III. Methodology

As it has been mentioned before, the objective of this analysis is to create a system that learns features that fail to discriminate between source and target distributions but that is good enough to learn the mapping between those distribution and their labels. The motivation is to implement this objective in the context of a Neural Network, whose hidden layer will be trained for a classification task and at the same time not be informative in regards to the domain of the input.

As it was mentioned before, one of the most important references to the development of the project is the model developed in [1]. This model uses a domain adversarial neural network (hereinafter, "DANN") instead of an evolutionary strategy to encode the loss function.

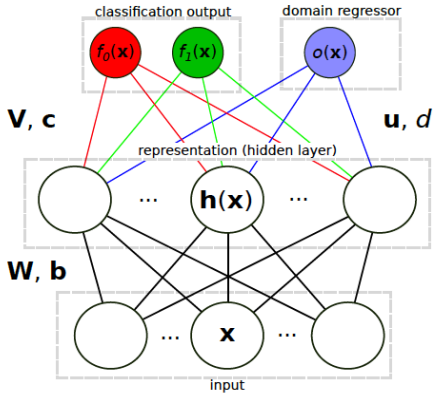

Fig. 3. DANN architecture[1]

The idea shown in the previous figure from the DANN approach in [1] is that the hidden layer $\mathbf{h}(\cdot)$ maps an example that can be from the source or from the target into a representation where the output layer $\mathbf{f}(\cdot)$ classifies correctly the source sample, while the domain regressor $o(\cdot)$ cannot detect if an example belongs to the source example or target example.

In comparison, the model proposed in this report have the same input but the representation will not be with a hidden layer. Instead, a neural network will be created but the output will be a set of features that will be used to train a Random Forest. Then, an evolutionary process will be used to adapt the weights of the neural network. This process is shown in the following figure.

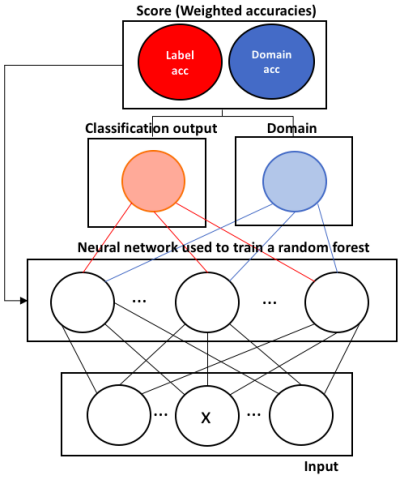

Fig. 4. Evolutionary Strategies for Domain Adaptation

A. Data sets

As it was mentioned before, the data sets that will be used are the MNIST-M data set and OFFICE-31 data set. The later one, OFFICE-31, is a benchmark for domain adaptation that encompass 4,100 images in 31 different classes or categories. These images were collected from three different domains:

- Amazon which contains images downloaded from `amazon.com` since it permits to access to a large amount of data relatively easy;
- Webcam that contains images taken by a web camera meaning that they have low resolution and show significant noise; and
- DSLR that contains images taken by a digital SLR camera with different photographically settings in realistic environment with natural lighting conditions.

This composition of data sets permits to review the adaptation of models learned in the web, which can be seen as a real life problem. On the other hand, domain transfer between high-resolution to low-resolution images allow for a very controlled research of category model adaptation as the same objects are recorded on both domains[13]. Therefore, several experiments will be performed by changing the source and target set.

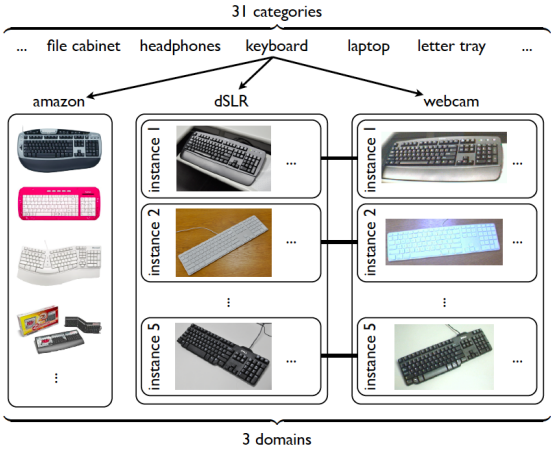

Fig. 5. Example of OFFICE-31[13]

In relation to the other data set, MNIST is a database of handwritten digits that was created by mixing several times the samples from the National Institute of Standards and Technology database (NIST). The MNIST database contains 60,000 training images and 10,000 testing images[10]. The MNIST[11] data set will be the source data set and in order to define the target data set, MNIST-M, the digits from the original set will be blended over patches randomly selected from color photos BSDS500[2]. Therefore, the difference between the source and target set will be that the lately will be digits blended digits over a non-uniform background. This will be done by generating an output sample produced by taking a patch from a photo and inverting its pixels at positions that correspond to the pixels of a digit. Formally speaking, this transformation is defined for to images I^1, I^2 as $I^{out}_{ijk} = |I^1_{ijk} - I^2_{ijk}|$, where i, j are coordinates of the pixel and k is a channel index.

An example of the MNIST-M is shown in the following picture:

Fig. 6. Example of MNIST-M[9]

IV. EXPERIMENTS

After the creation of the source and target set for each data set and for the correspondent experiment, the following steps were implemented in order to perform the analysis of the application of evolutionary strategies for domain adaptation.

1) The first step was to create an neural network that takes as input the data from each data sets (MNIST-M and OFFICE-31) and generates as an output a set of specific features. Rectified linear units were incorporated within the creation of the neural network because during the training phase, they help to have a more efficient process even in cases where pre-training was not applied. The model implemented was the following:

```
model = Sequential()
model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu',
                 input_shape = self.input_shape))
model.add(MaxPooling2D())
model.add(Conv2D(48, kernel_size = (3, 3), activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(8, activation = 'relu'))
```

2) Then, the outputs from the random neural network were used to train a Random Forest. Random forests can help to ensure a sufficient number of set of features (outputs from the neural network) and reduce over-fitting as it was mentioned before. In this case, the number of trees used in the forest were 10.
3) The next step was to create an evolutionary process using SNES in order to adapt the weights of the Neural Network. It is important to notice that the score of the classifier will take into account domain adaptation. The score that will be given back to SNES will be a weighted sum between the performance of a random forest (how bad it discriminated between source and target domain) and at the same time how well it discriminate between the different classes. The score that was returned to the SNES was the following[1]:
   - For the MNIST-M, the formula was the following:

$$score = 0.95 \times label + 0.05 \times domain$$

   - For the OFFICE-31, the formula was the following:

$$score = 0.45 \times label + 0.55 \times domain$$

The weights of the score were determined by running several trials in order to reflect the objective of penalize the result if the model distinguished correctly between source and target domain and reward it if the model classified correctly the labels.
On the other hand, the number of the different set of weights that should return the SNES for evaluation are the same for both data sets. The definition of this number was made by running the model several time on a small sample of the data set MNIST-M and its results were incorporated in the data set OFFICE-31.

Dataset	Generation	Population size
MNIST-M	30	10
OFFICE-31	35	10

TABLE I
PARAMETER FOR SNES

4) Finally, a plot of the loss obtained at each generation in both sets was constructed. This helped to determine the number of generation and the result for each graph are discussed in the following section.

The output of the model for each data set will be the accuracy of the prediction of labels in source validation set, the accuracy of the prediction of labels in target validation set and the accuracy of the prediction of domain in a validation set that contains observations from the source and target domain. It is important to mention that for the systems for the two data sets, the model for the label classifier is trained using only the source data. On the other hand, the model for the domain classifier is trained using a combined set

[1]The model and code implemented were take from `https://github.com/ssamot/RL_notebook` by the authorization of Spyros Samothrakis.

that includes observations from the source and target domain.

V. Discussion

A. Accuracy of the system

As it is described in the previous sections, the results from the proposed model were compared against other studies, specifically with [9]. The results from the proposed system are the resulted accuracy from the label classification using a model trained with the source domain data and tested with the a data set from the target domain set. Also, the results include the resulted accuracy from the domain classification for which the system was trained using a set that included observations from the source and target domain and tested on a combined validation set that was not used during the training phase.

We would like to have a system were the accuracy of label classification is as high as possible for the testing on the target domain. Also we would expect a high accuracy value for the testing on the source domain since the model is trained in said set. Finally, we would also expect to have a value around 0.5 for the domain accuracy since this would mean that the system cannot distinguish between source and target domain and it is classifying just by chance.

For the MNIST-M set, the source domain is the MNIST set set and the target domain is the MNIST-M set. For the OFFICE-31 set, three experiments were performed: (1) The Amazon set was considered as source domain and DSLR set as target domain; (2) The DSRL set was considered as source domain and Webcam set as target domain; and (3) The Webcam set was considered as source domain and DSRL set as target domain.

The resulted accuracies from the previous experiments are shown above where the following notation is used:

- $Accuracy_{Domain}$ is the domain accuracy;
- $Accuracy_{Target}$ is the accuracy of the classes predictions in the target validation set;
- $Accuracy_{Source}$ is the accuracy of the classes predictions in the source validation set;
- $Accuracy_{Ganin}$ is the accuracy showed in [9].

Method *Source* / *Target*	MNIST / MNIST-M
$Accuracy_{Domain}$	0.4853
$Accuracy_{Target}$	0.1773
$Accuracy_{Source}$	0.1574
$Accuracy_{Ganin}$	0.7666

TABLE II
ACCURACY RESULTS ON MNIST-M SET

For the MNIST-M set the system do not distinguish between the source and target domain (the value of the accuracy is around 0.5). This is one of the desired objectives for the system, nevertheless, the performance of the system is quite poorly to classify the labels of the examples for the target and source domain.

In the case of the OFFICE-31 set, there will be different evaluations where the source and target data will be different as it is shown in the following table.

Method *Source* / *Target*	Amazon / Webcam	DSLR / Webcam	Webcam / DSLR
$Accuracy_{Domain}$	0.9938	0.9381	0.9362
$Accuracy_{Target}$	0.0222	0.0616	0.0667
$Accuracy_{Source}$	0.9799	0.1850	0.3808
$Accuracy_{Ganin}$	0.730	0.964	0.992

TABLE III
EXPERIMENTS ON OFFICE-31 SET

From the results showed in the previous table, the accuracies do not have the desired values for any of the scenarios. In the case of the domain classifier, the accuracy for each of the scenarios is around 0.9 meaning that the system is learning to make a discrimination between the source and target domain. On the other hand, the scenarios where the source domain is the DSLR set or the Webcam set, the results when the model is tested in the source domain are low in comparison to the scenario where the Amazon set is the source domain. This can be explain by the fact that the resolution and quality of the images in the first mentioned data set are considerably low and/or there is noise in the images that do not let to train the correctly the model. This problem is more obvious when the system is tested on observations of the target domain having as a result a low accuracy for the prediction of the labels.

B. *Analysis of loss plots - SNES*

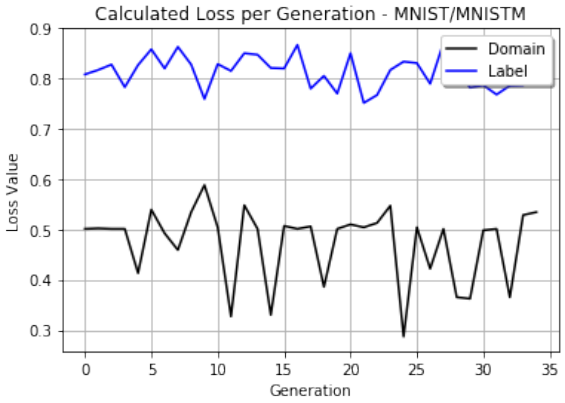

Fig. 7. Plot of loss against generation SNES - MNIST-M

The plot of the MNIST-M set shows relevant information for the loss of the domain classification and label classification. For the model analyzed, it seems that the loss for each classifier fluctuates around certain values so there cannot be seen any decrease of this value. Also, it can be seen that for the domain classification, the system since the first generation fluctuates around 0.5 which is a value that we would like to see in the final model.

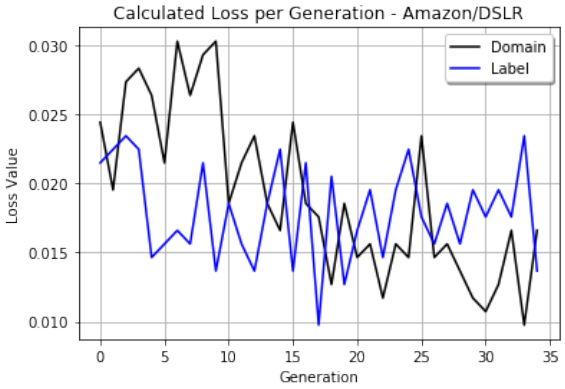

Fig. 8. Plot of loss against generation SNES - OFFICE-31 (Source: Amazon data set, Target: DSRL data set)

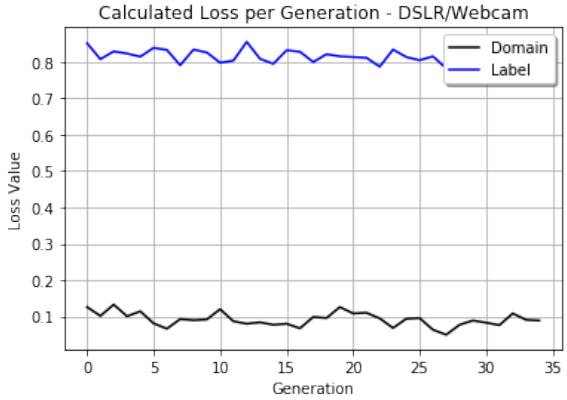

Fig. 9. Plot of loss against generation SNES - OFFICE-31 (Source: DSLR data set, Target: Webcam data set)

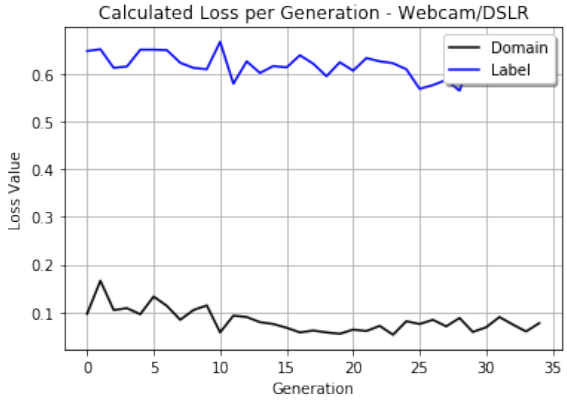

Fig. 10. Plot of loss against generation SNES - OFFICE-31 (Source: Webcam data set, Target: DSRL data set)

The plots of the OFFICE-31 reflect different scenarios. The first plot that is the scenario where the source domain is the Amazon set and target domain is DSLR, shows that the loss value is low as desired for the label classification, but the loss value for the domain classifier is not close to the desired value. Finally, we can see a decrease in the loss value with older generations but this decline is not fast as we would like to have.

VI. CONCLUSION

The way a useful domain adaptation system should perform is by minimizing the loss of the label classifier and maximizing the loss of domain classifier for the parameter of the underlying deep feature mapping as to minimize the error on the training set for the label classification. The purpose of this report was to investigate a system that started by creating a Neural Network where the input was the data and outputs a set of features that were then used to train a Random Forest.

The novel about the proposed system was that the weights of the neural network were constantly updated by applying an evolutionary process using a Separable Natural Evolution Strategies to adapt the weights of the Neural Network. The score used to start the SNES measured how well the Random Forest was able to discriminate between the classes and how bad it was to discriminate between sources. Several experiments were performed in two different data sets: MNIST-M and OFFICE-31.

The expectation of including the SNES in the process was to improve the performance of the system but there is opportunity to include some improvements for the score function that is returned. Since it is defined as a weighted sum, the weights were determined empirically so other values or calculations of weights may improve the performance of the system. Furthermore, in the case of the OFFICE-31 and given the results of the systems, this approach could be explored more deeply.

Even that the proposed system is an embed structure with three training process: incorporate feature learning, domain adaptation and classifier learning, for the Neural Network model and the analyzed data sets, it does not have the desired results. As it was described in the previous section, according to the parameters and tuned defined in these experiments, the results have a possibility to improvement. In the case of the OFFICE-31 the systems performs poorly in terms of the discrimination of the domain. The accuracy in this case is too high (around 0.9 every generation) and this way, the system is learning the differences between the domain getting farther from the objective. For the data set MNIST-M, the final system provides good result for the discrimination of the domain given that the accuracy value is close to 0.5, nevertheless, for the label classification the results are poor and do not provide a feasible application.

A possible future discussion could be the comparison with a model where the system for the label classifier is trained on a combined data set that includes observations from both data sets. This will mean to include some observations from the target domain with their correspondent labels. This can be an obstacle to the application since in real life and for the situations where Domain Adaptation can help, the premise is that acquiring labels from the target domain can be difficult. Other future discussion about the model can be hyperparameter tuning. This can be performed (for the Neural Network, Random Forest, etc.) buy it is important to keep in my mind that this process can be exhausting and long. Nevertheless, given the results of the model, this can be incorporated in future models.

In order to review the model and results described before, please refer to `https://github.com/tanyaramirezf/CE888_Assignment`. As it is mentioned there, in order to replicate the model in both data sets, OFFICE-31 and MNIST-M, certain data sets must be downloaded before. It is important to download the sets locally in order to create the correspondent source and target domain data sets and replicate the proposed model.

REFERENCES

[1] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[2] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[3] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[4] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[5] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.

[6] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[7] Ali Farhadi and Mostafa Kamali Tabrizi. Learning to recognize activities from the wrong view point. In *European conference on computer vision*, pages 154–166. Springer, 2008.

[8] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[9] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[10] Ernst Kussul and Tatiana Baidyk. Improved method of handwritten digit recognition tested on mnist database. *Image and Vision Computing*, 22(12):971–981, 2004.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[13] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

[14] Tom Schaul, Tobias Glasmachers, and Jürgen Schmidhuber. High dimensions and heavy tails for natural evolution strategies. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 845–852. ACM, 2011.

[15] Sanatan Sukhija, Narayanan Chatapuram Krishnan, and Gurkanwal Singh. Supervised heterogeneous domain adaptation via random forests. In *IJCAI*, pages 2039–2045, 2016.

[16] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011.

[17] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3381–3387. IEEE, 2008.