Report on independent study research at MESL Lab for the Winter Quarter 2020 - MGT 499

Domain Adaptation on Scrabble Using Transfer Learning Techniques

Under the guidance of -

Jason Koh

MESL, CSE Department

University of California, San Diego

jbkoh@eng.ucsd.edu

Submitted by -

Ashish Gupta

MSBA, Rady School of Management

University of California, San Diego

asg007@ucsd.edu

Problem Statement

The Scrabble framework provides a versatile approach to map building information in the form of thousands of data points on to a common schema, successfully retrieving semantic metadata from unstructured raw metadata. This is done while reusing known information in existing buildings to reduce the amount of effort for domain experts to provide input labels. Modern building infrastructure is supported using sensors and networked into a Building Management System(BMS). BMSes manages data from sensors such as room temperature, power meter; from actuators such as dampers for control of air flow, fans for exhaust; and from configurations such as the temperature setpoint for heating. Brick is a metadata schema which provides for a complete vocabularies and relationships to describe such resources in a machine readable format, thus enabling portability of applications. Scrabble is currently implemented and evaluated for five buildings using Brick as the target schema.

The high-level objective through this research is to obtain a more comprehensive interoperability in internet of things by providing for a robust scalable implementation of Scrabble framework. Best deep networks and architectures are trained on massive amounts of data that are labeled. In the absence of labeled data for a certain classification task, domain adaptation is usually an attractive method given that labeled data of similar nature but from a different domain are available. The intention is to scale the framework to multiple buildings, each of which would correspond to a domain shift or dataset bias, by reducing the adaptation reduce the difference between the training and test domain distributions and thus improving generalization performance of the framework. Given the labeled source building data and the target building input data points, the purpose is to obtain the predictions for the target building labels. With more buildings under the umbrella of trained domain distributions, a better unified metadata schema for building information could be achieved. The goal of domain adaptation is to eliminate or reduce the mismatch between the training data and the test data. This research attempts to define a new paradigm for a progressive learning of labelling tasks with concept of domain adaptation.

Domain Adaptation in Scrabble

Scrabble uses an active learning framework for normalising the metadata of sensors in multiple buildings by using a transferrable intermediate layer. Scrabble makes use of machine learning algorithm called Conditional Random Field (CRF) Classifier to map a sentence from target building to a set of Brick tags. This is trained with the help of source samples or examples from domain experts. These Brick tags form a reusable Intermediate Representation (IR) for effectiveness in mapping it to the final labels.. A multi-Layer Perceptron(MLP) is used to map the Tags to the corresponding Tagsets. Scrabble iterates through all the target building data points until it identifies all building labels with high confidence.

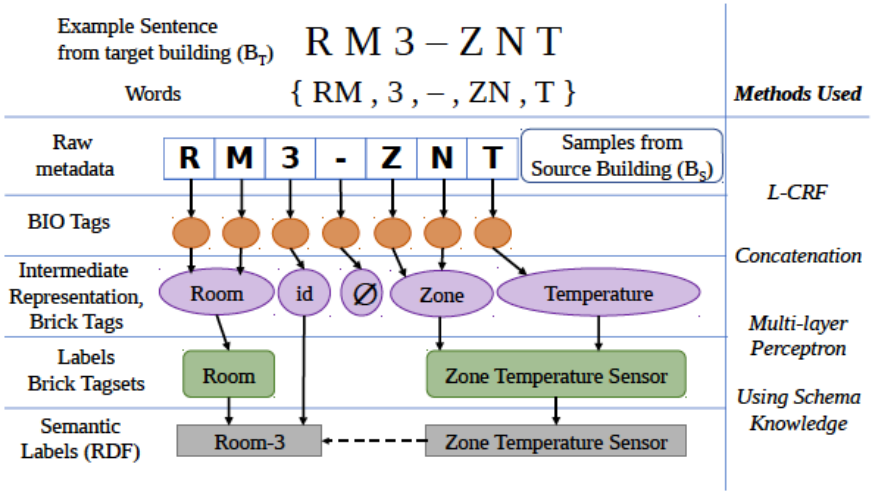

Fig A. Data mapping from raw metadata to target schema

Given this underlying hierarchical structure of Scrabble, domain adaptation can be implemented in two specific layers. The CRF layer when the tokens from a sentence are converted to Tags or in the MLP layer to convert the Intermediate Representation (IR) tags to the final tagsets. In this research, we attempt on implementing domain adaptation in the latter by obtaining training or source building information and using transfer learning techniques to map the tags from target building to the tagsets despite the shift in the domain.

Methodology

This study attempts to validate the usage of promising methods currently in on-going research in the field of domain adaptation. We focus on two main methods for the implementation - Neural Adapter and Adversarial Domain Adaptation.

A. Neural Adapter

This method aims at transferring knowledge in an incremental progressive way within the same domain rather than to the other domains. It is assumed that the target output domain is included in the source output domain. A neural adapter is used to mitigate the differences between the source and target label distributions and learn from them. This provides additional important information to the final model. In the Scrabble context, we would use the source building tags to tagset conversion information and transfer the knowledge to a target building where new label categories appear. Following is a detailed explanation on the progressive transfer learning approach.

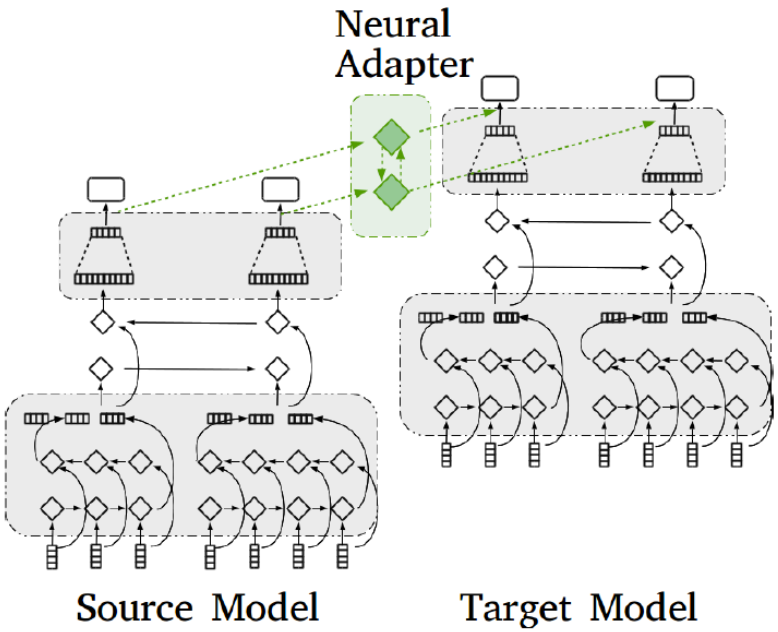

Fig B. Implementation Architecture

i. Training of a source model

A labeling model *Ms* is trained on a source dataset *Ds* to obtain optimal parameters These are saved and reused for transfer learning. We use a multi-layer perceptron to train this model.

ii. Parameter Transfer

The parameters in the output layer is initialized with the weights drawn from the normal distribution of the saved weights from the pre-trained source model Ms, taking the mean and standard deviation of the values. The weights not in the output layer are set with the same values as obtained from the optimum weights from *Ms*. We test this condition by implementing various other combinations of weight assignment for parameter transfer as explained in the future sections.

iii. Target model training

Here, we use the transferred parameters and train the target training data by designing an MLP and using this MLP along with the fixed source model to obtain multi-label classification predictions on the input. Here, we introduce an extra layer of MLP between the source and the target MLP outputs, called the neural adapter. This component solves the disagreement in annotations between the source and target data. This layer helps to map the predictions from the output space of the source domain to that of the target domain. We obtain the element-wise summation of the output from fully connected source layer and the extra layer to that of the fully connected target output layer, following with the output consists in a softmax to obtain the final result.

B. Adversarial Domain Adaptation

Adversarial adaptation methods have become increasingly popular approach to minimise a domain discrepancy with respect to a domain discriminator. This is closely related to the generative adversarial learning or GANs where two networks generator and discriminator are pit against each other and both try to outperform the other, hence reaching a state of domain generalisation that can result in better target scope predictions. This works under the assumption that probability distribution of source domain is not equal to the probability distribution of the target domain but the conditional probability distribution of the labels given an instance from both the domains are the same.

The overview of this method is to perform some kind of transformation on bot source and target domain to get the distributions closer. The classifier is trained on the transformed source domain and since both the distributions are now on the same scale, the model obtains better results on target domain. Assume the neural network used for transformation is denoted by ‘T’ with the parameters ‘P’. The instances from source and target domain are denoted as ’s’ and ’t’. The transformed vector is denoted as Vs and Vt for source and target.

Then,

$$T(s,P) = Vs \text{ and } T(t,P) = Vt$$

The objective is to obtain the probability distributions of Vs and Vt closer to each other.

$$P(Vs) = P(Vt)$$

This method comprises of the following components.

i. Feature Extractor

This MLP performs transformation on the source and target distribution

ii. Label Classifier

This MLP network performs classification on the source domain data points and predicts the labels.

iii. Domain Classifier

This MLP predicts whether the output of the feature extractor is from the source or target distribution. The idea is that it classifies the domain of the transformed input.

The intuition is that the feature extractor produces a type of transformation that domain classifier is not able to classify the domain of the transformed instances. This is done by training the feature extractor and domain extractor in a way that feature extractor is trained to maximize the domain classification loss, while the domain classifier is trained to minimize the domain classification loss. The label predictor is trained to predict the labels on the transformed instances since the source is labeled. The feature extractor thus learns to produce discriminative and domain-invariant features.

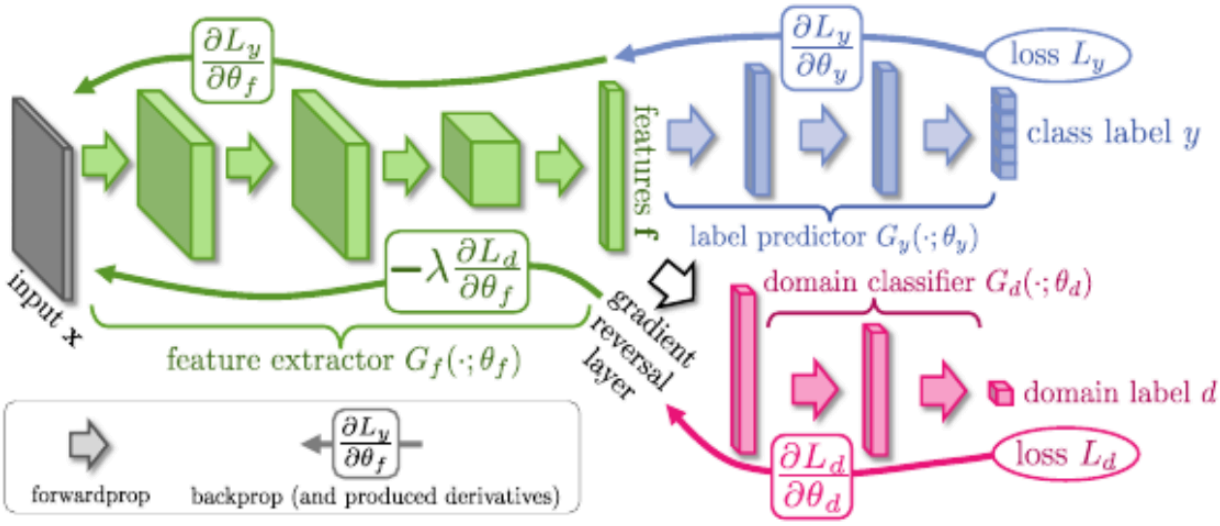

Fig C. Adversarial Domain Adaptation architecture
Source: https://arxiv.org/pdf/1409.7495.pdf

We place a gradient reversal layer between the feature extractor and the domain classifier which acts as an identity function during forward propagation but during back propagation it multiplies the input by -1. Intuitively, this layer is leading to the opposite of gradient descent, that is, it is performing gradient ascent on the feature extractor with respect to the classification loss on the domain predictor.

Implementation and Results

These two methods - Neural Adapter and Adversarial Domain Adaptation are used in the Scrabble context to perform domain adaptation. Once we obtain the tags from the sentences from the target building data points, we perform domain adaptation in the second stage of the framework where the Intermediate Representation (IR) Tags are converted to the Tagset labels. We use samples from both source and target building to form the training data, to incorporate both the domains for the training process. The datapoints are vectorised to a csr matrix using CountVectorizer, including the response variable which is the tagset label. The domain index of the training data is also obtained and fed into the modelling process. Multiple variations and combinations are tried and tested to achieve better results.

A. Neural Adapter method with fixed source MLP and adapter layer

In this case, the source model in the neural adapter architecture is pre-trained to predict the labels. For the target model, the weights are initialized by taking the normal distribution using the mean and standard deviation obtained from pre-trained source weights. The source layers are then fixed and made non-trainable. An adapter layer is added to connect the output of the source and the target layers, which is also initialized in a similar fashion. Since this is a multi-label classification task, the loss for all the MLP models is obtained by 'binary-crossentropy', using 'relu' activation and 'rmsprop' as the optimization algorithm. The final predictions consist in a softmax at the output layer. The training occurs over 100 epochs and we use 300 samples from the source building and 200 samples from target building to form the training input points.

B. Neural Adapter method with fixed source MLP and no adapter layer

In this case also, the source model in the neural adapter architecture is pre-trained to predict the labels. For the target model, the weights are initialized by taking the normal distribution using the mean and standard deviation obtained from pre-trained source weights. The source layers are then fixed and made non-trainable. The difference is that a direct element-wise summation is performed between the source and target output layers without the adapter MLP. Since this is a multi-label classification task, the loss for all the MLP models is obtained by 'binary-crossentropy', using 'relu' activation and 'rmsprop' as the optimization algorithm. The training occurs over 100 epochs and we use 400 samples from the source building and 300 samples from target building to form the training input points.

C. Neural Adapter method with non-fixed source MLP and no adapter layer

The source model in the neural adapter architecture is pre-trained to predict the labels. For the target model, the weights are initialized by taking the normal distribution using the mean and standard deviation obtained from pre-trained source weights. The difference here is that the source layers are not fixed and made trainable. A direct element-wise summation is performed between the source and target output layers without the adapter MLP. Since this is a multi-label classification task, the loss for all the MLP models is obtained by 'binary-crossentropy', using 'relu' activation and 'rmsprop' as the optimization algorithm. The training occurs over 100 epochs and we use 400 samples from the source building and 300 samples from target building to form the training input points.

D. Adversarial Domain Adaptation

For this implementation, functional keras was used to build the three separate MLP models feature extractor, label classifier and the domain classifier. The output of feature extractor was fed as inputs to both label classifier and the domain classifier. Gradient reversal layer was used in the domain classifier MLP. Over 200 epochs was used to train the model and obtain the domain classification and label classification accuracy. The loss used was 'binary-crossentropy' with 'relu' activation and 'rmsporp' as the optimisation algorithm. The training was performed in batches and the model performance was evaluated.

Evaluation of the performance of a model was done using set accuracy since that this was a multi-label classification with a lot of sparse values in the output. A multi-layer perceptron with two hidden layers to classify labels was considered as a baseline model for comparison. This model was trained without any sort of domain adaptation.

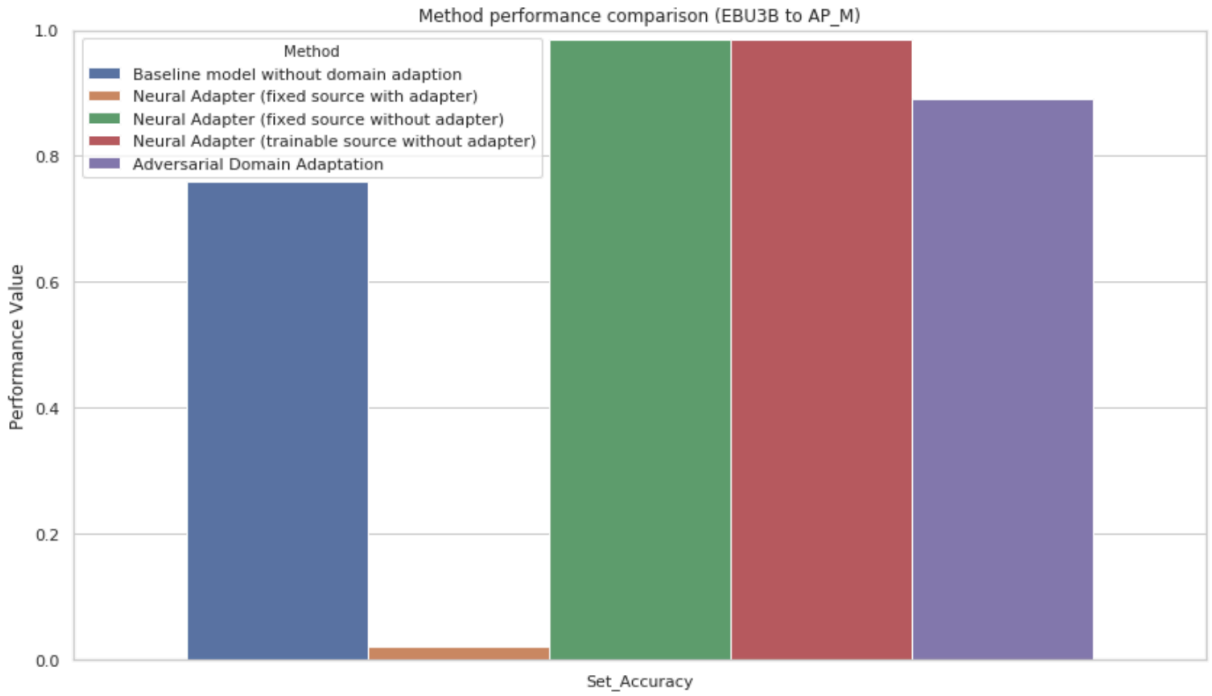

Fig D. Set Accuracy for all Methods using 'EBU3B' as Source building and 'AP_M' as Target Building

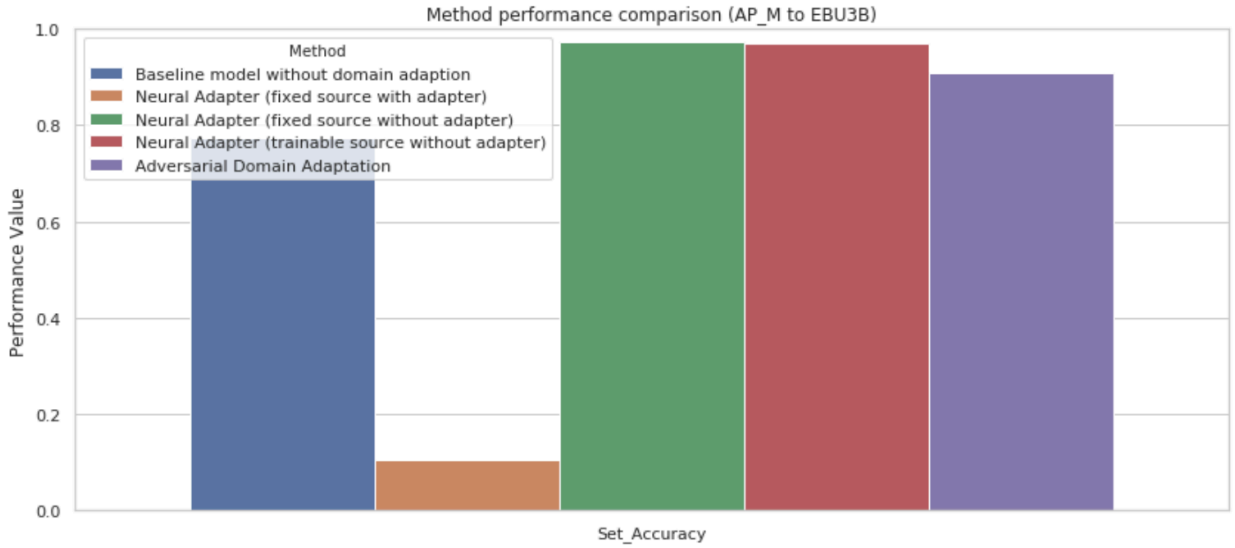

Fig E. Set Accuracy for all Methods using 'AP_M' as Source building and 'EBU3B' as Target Building

The set accuracy obtained by these models were compared with the baseline model implemented without any adaptation technique. Both the directions were considered, meaning, the source and target buildings were reversed, and the results turned out to almost equivalent. The neural adapter implementation without the adapter layer outperformed the adversarial domain adaptation. Both these models had a higher set accuracy than the baseline model. The above methods suggest the comparison of the results with other domain adaption techniques by using the source data in the native model. The following chart displays the performance comparison when the native model was trained without the source data.

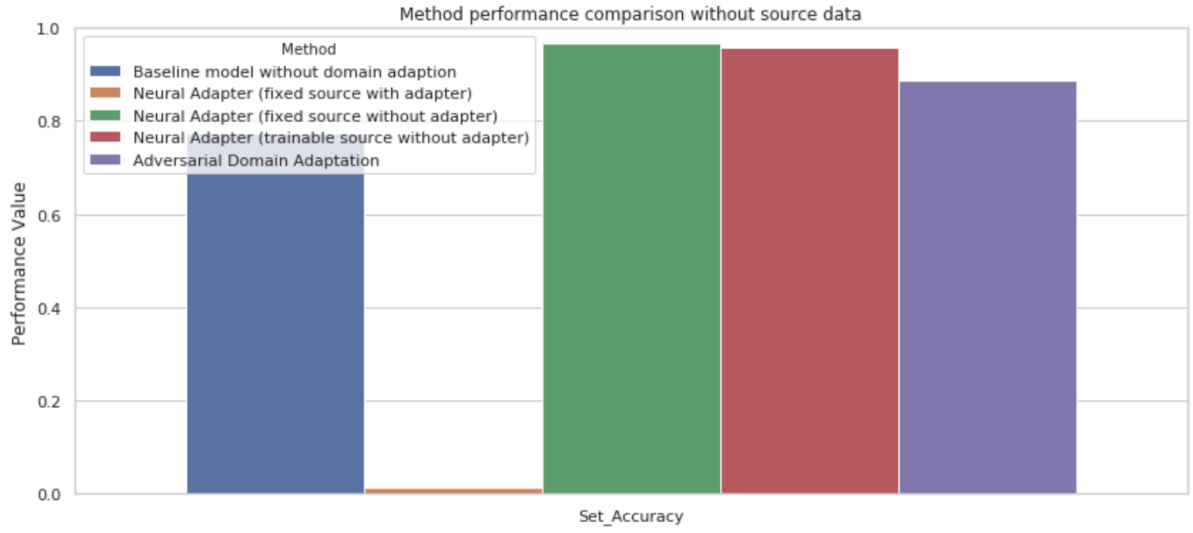

Fig F. Set Accuracy for all Methods training the native model without source data

Clearly, if the source data is not used to train the native model, we see a slightly lower set accuracy for almost all the methods, but the distribution of the results remains the same. This is done to check the transferability of the domain adaptation techniques.

Listed below are some of the inferences from the results obtained -

- The results were obtained by checking for two domain shifts in buildings ‘AP_M’ and ‘EBU3B’. The set accuracy was consistent in both these transformations.
- It is observed that the transfer learning method without using the adapter layer in the output but using pre-trained source to initialise the weights performed the best.
- The performance did not vary much if the source layer was made trainable or non-trainable.
- When the adapter MLP was added to both the source output and the target output layer, we notice a drastic decrease in performance. This may be due to source and training both training independently and the adapter layer being biased to zero values, leading to erratic predictions.
- The adversarial domain adaptation using functional keras performs well in discriminating the domain with 0.5 accuracy and the labels with greater than 0.8 accuracy.
- One of the reasons the adversarial domain adaptation did not work the best is probably that the gradient reversal layers leads to vanishing gradient problem once the domain predictor has achieved maximum accuracy.
- Another reason might be that same weights are being used for performing the transformation on the source domain and the target domain. Since, both source and target domain might have different features therefore, shared weights might lead to less number of parameters for learning independent features and transforming both the distributions.

Business Applications of Domain Adaptation

Majority of existing domain adaptation methods makes an assumption of freely available source domain data. An equal access to both source and target data makes it possible to measure the discrepancy between their distributions and to build transformations common to both target and source domains. Since source data are routinely a subject of legal constraints between data owners and data customers, in reality this assumption rarely holds. When source domain data can not be accessed, decision making procedures are often available for adaptation nevertheless. Following are some of the applications in the business context that can be accomplished using Domain Adaptation.

- These techniques could be useful in digital advertisements, where partners/sellers contact potentially interested online users with their products. Within the digital advertisement domain, there are multiple platforms like user re-targeting or prospecting. Prospecting basically means targeting un-tapped user-base. One of the challenge with prospecting is partner-cold start problem, which is on-boarding a new partner to the prospecting platform. Domain adaptation could be used to solve this problem by leveraging the large amount of re-targeting data and transferring the model results to the prospecting domain.

- Domain Adaption has a great potential in visual classification tasks. Using Convoluted Neural Networks(CNNs) with transfer learning techniques can result in robust multi-domain image classifications and result in better classification. One of the examples for such an implementation could be applied in e-commerce websites to classify product images to specific sections or sub-sections.

- Deep fake, which is an infamously hot topic currently use Generative Adversarial Networks to create domain shifts/noise in a real image. Domain adaptation techniques can be used to classify a real image from a fake one and prevent massive misuse of this technology.

References

[1] Zodiac:
https://www.synergylabs.org/yuvraj/docs/Balaji_BuildSys15_Zodiac.pdf

[2] Scrabble:
https://mesl.ucsd.edu/pubs/Jason_BuildSys2018Scrabble.pdf

[3] Brick:
https://www.topas-eeb.eu/images/2016_Brick_BuildSys_UnifiedMetadata.pdf

[4] Adversarial Discriminative Domain Adaptation:
http://openaccess.thecvf.com/content_cvpr_2017/papers/Tzeng_Adversarial_Discriminative_Domain_CVPR_2017_paper.pdf

[5] Neural Adapter:
https://www.aaai.org/Papers/AAAI/2019/AAAI-ChenLingzhen.6418.pdf

[6] Multi-Domain Adversarial Learning:
https://arxiv.org/pdf/1903.09239

[7] Unsupervised Domain Adaptation by Back propagation:
https://arxiv.org/pdf/1409.7495.pdf