2018 IEEE International Conference on Information Reuse and Integration for Data Science

Deep Transfer Learning for Traffic Sign Recognition

Grant Rosario, Thomas Sonderman, and Xingquan Zhu
Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University
Boca Raton, FL 33431, USA
{grosario2015, tsonderm, xzhu3}@fau.edu;

***Abstract*—In this paper, we study using deep transfer learning to utilize knowledge gained from an already developed, large dataset of traffic signs of a specific country/region, and use this knowledge to better recognize traffic signs from another country/region, under a deep learning framework. This provides the possibility of using an already established dataset from other regions to aid in the recognition of a desired target dataset, freeing users from the burden of data gathering and labeling. We propose three deep transfer learning methods, and two of them demonstrate significantly improved accuracy compared to the simple deep learning classifier. This research shows transferring knowledge between deep learning classifiers can provide higher accuracy for traffic sign recognition than a model which implements only deep learning to recognize traffic signs.**

I. Introduction

The area of traffic sign recognition (TSR) has been a target of intense research, in part because it branches from the broader field of study of object detection, but also due to its many complexities and obstacles such as motion blur, occlusion, color distortion, etc [1]. While these challenges make the problem of TSR intriguing, perhaps more justification for researching TSR lies in its potential benefits.

The ability to observe and adhere to traffic signs is essential for safe and defensive driving patterns and thus, the systematic ability to detect and recognize traffic signs automatically via a front-facing camera could provide highly beneficial information to the driver in times when distractions arise, signs are out of view, or the field of view is diminished due to increases in speed or other high stress situations [2]. This research aims to provide new insights into artificial intelligence (AI) applications including deep learning and transfer learning for the purposes of traffic sign recognition.

Just like humans use their eyes and brains to visually observe and extract information from the world around them, the goal of computer vision (CV) is to enable computers and machines to utilize cameras and processing power to not only see, but extract information from images or videos equal to or better than human ability. This field can range from CV tasks in agriculture, medicine, facial recognition, and many others. Due to this breadth, the task of TSR can be thought of as one big CV problem with some of the best solutions currently being delivered via machine learning techniques.

A. Deep Learning

Deep learning (DL) is a branch of machine learning which comes from the notion of teaching a computer extremely complex concepts by first teaching it simple ones in a hierarchical manner, thus leading to a deep graph of several layers. A convolutional neural network (CNN) is the typical type of networks used in computer vision applications. These types of networks specialize in processing data with a grid-like topology such as image data, which is a two-dimensional grid of pixels. CNNs also commonly utilize some sort of pooling to shrink the spatial dimensions of the image into more detailed convolutions as well as parameter sharing so the model knows to apply learned features to the entire image. A typical CNN process begins with reading in an image, typically a 3-channel RGB image, and utilizing a filter, also called a kernel, to scan the image and learn significant features of the image via this filter. Along with the typical hyper-parameters found in a standard deep neural network (DNN), CNNs bring with them additional ones such as stride, padding, filter size and a few others.

B. Transfer Learning

One of the milestones which led to the current success of DL was the development and continuous growth of research datasets. Not only do these datasets provide researchers with a solid foundation for performing experiments in AI and, specifically, DL, but they also free up experimenters from having to gather a large enough amount of data to provide meaningful results. Thus, convolutional neural networks are rarely trained from scratch but are usually trained using one of the many massive data sets that have been developed. Training a network on a certain dataset and applying the knowledge learned from this training to a different but related dataset is a concept known as transfer learning (TL) and is an important area of deep learning research [3].

Typically, when training on datasets, there are four guidelines for when and how to apply TL based on the size of the new dataset and how similar it is to the original dataset [4].

- *Small target dataset similar to the auxiliary dataset*
  In this case, the recommended method is to cut off the last layer of network, add a new fully connected layer to the end that matches the number of classes in the new dataset, randomize the weights of the new fully connected layer while freezing the weights from the pre-trained network and use the updated weights of the fully connected layer to train the network with the new data.
- *large target dataset similar to the auxiliary dataset*
  The method in this situation is the exact same as if the dataset were small, the only difference being that freezing the weights is not recommended. Rather, they should be initialized to the values of the pre-trained weights and trained again using the new data.
- *Small targetdataset different from the auxiliary dataset*
  In this scenario, we should slice off most of the pre-trained layers at the beginning of the network, add a new fully-connected layer with the appropriate number of classes, freeze the weights of the remaining pre-trained layers and randomize the fully connected layer, using its weights to train the network.
- *Large target dataset different from auxiliary dataset*
  In this specific case, we could train the entire network from scratch but it would still be recommended to initialize the weights with the pre-trained weights, similar to what the recommendation is for when the data is large and similar.

Most modern available datasets which are adequate for TSR, such as GTSRB, are based in countries outside the U.S [5], [6], [7], [8]. Due to this, it can be very difficult to build an entire deep learning model from scratch consisting of thousands of U.S. signs. Fortunately, this is not entirely necessary thanks to transfer learning. Similar to how learning a second language is easier than learning the first and driving a truck is simpler when one knows how to drive a car, TL consists of the idea that a model can more easily learn a set

978-1-5386-2659-7/18/$31.00 ©2018 IEEE
DOI 10.1109/IRI.2018.00034

of different features if it has already learned a similar set of features [9].

In 2014, Razavian et al. achieved consistently superior results from training a linear SVM classifier using 4,096 features extracted from a layer in the OverFeat CNN, rather than developing a system from scratch [10], [11]. Also in 2014, Yosenski et al. analyzed and discussed the possible benefits of transferring features from a pre-trained DNN and found that initializing a network with transferred, pre-trained features can almost always provide a boost in generalization [12]. The functionality of TL has been under experimentation recently with Zhao et al. publishing a paper proposing the idea of utilizing a pre-trained feature descriptor, training a multi-layer perceptron to also generate a feature descriptor, and finally concatenating both descriptors to improve efficiency with little to no cost in accuracy [13]. These works support the idea that TL, while still in its early experimentation stages, has the potential to provide massive benefits to TSR tasks and AI problems as a whole.

While TSR is a popular research area, most of it focuses on new deep learning models and recognition speed. We believe our research is rather unique in that we focus on the problem of gathering datasets and how already existing datasets can benefit potential new datasets. Specifically, the scenario with a small target dataset similar to the larger auxiliary dataset.

II. TOLY: Target Only Deep Learning for Traffic Sign Recognition

We begin TSR by introducing a baseline which utilize target only (TOLY) classification. This process signifies a model using the same dataset for auxiliary and target data, meaning it is trained and tested from the same data without using data from other sources.

A. Preprocessing

Many traffic signs share the same color scheme, therefore, when it comes to recognition, the details of the sign graphics are of much higher significance than the color of the sign. Thus, we begin by converting the image to grayscale in order to emphasize the contrast of the graphics on the sign. Then, we apply a slight gaussian blur to increase the image clarity and we equalize the histogram to normalize the image. Lastly, we shuffled the training set which is a crucial step for training any CNN as it prevents the network from learning any sort of pattern in the data.

B. TOLY Architectures

In order to develop a substantial TOLY model, we develop a series of 10 convolutional neural networks to train the traffic sign data on. Each of these networks vary in their depth which corresponds to the amount of weight convolution layers they possess. The network architectures are implemented in a similar manner as our detection networks in regards to utilizing max-pooling, a fully connected feature vector, and a soft-max function for the final predictions, but they differ slightly in regards to convolution sizes. Firstly, because we are working with a grayscale image as our input, the initial input depth is 1. Each TOLY CNN's first convolution produces a depth of 3, then 6, then 9, with the depths increasing up to 24 depending on the amount of layers in the network. Every TOLY CNN also utilizes a filter size of 3, other than the 9-1x1-layer net which uses a few 1x1 filters. It is also important to note that our method takes into account the ability to add any amount of labels for prediction after the fully connected layer. Figure 1 provides a visualization of a 4-layer CNN in order to provide a better understanding of how these networks are implemented.

C. Overfitting

In order to combat overfitting, we implement the following techniques in each of our network architectures throughout this experiment:

L2 Regularization: The objective of this regularization technique is to add the L2 norm of the network weights to the network loss and multiply this by a small constant. This technique ultimately penalizes large weights, thus smoothing out the overall loss [14]. The formula we used for L2 regularization is

$$L' = L + \beta \frac{1}{2} ||\omega||_2^2 \tag{1}$$

where L is our loss, β is a hyper-parameter set to 0.001, and $\frac{1}{2}||\omega||_2^2$ is our L2 norm calculated as we train the network.

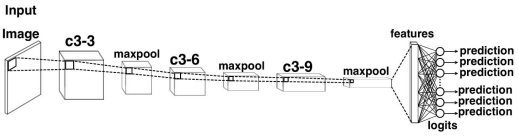

Fig. 1: Visualization of the 4-layer CNN used in the TOLY classification model. *c3-3* signifies a convolution of 3 layers using a 3x3 filter. *c3-6* signifies a convolution of 6 layers using a 3x3 filter. *c3-9* signifies a convolution of 9 layers using a 3x3 filter. The CNN finishes by flattening the *c3-9* convolution (after max-pooling) into a single-layer feature vector, generates logits via matrix multiplication and computes final predictions on the target data via a softmax function.

Learning Rate Decay: By slowly decreasing the learning rate of our training we decrease the effect of divergence and increase the speed of convergence. Although this concept does not specifically address overfitting like other techniques, it does provide better generalization for our learning. The formula we used for our networks is

$$\alpha = \frac{1}{1 + d * i} * \alpha_0 \tag{2}$$

where d is our decay rate hyper-parameter set to 0.0001, i is the current epoch number, and α_0 is our beginning learning rate, which we initialize to 0.001.

Early Stopping: We implemented one of the most successful techniques for combating overfitting, which is to force a network to stop training if it has not gained beneficial knowledge after a certain period [15]. Our technique keeps track of the validation accuracy of our networks, only saving the model if the validation accuracy has improved and stops training if this validation accuracy has not improved in 25 epochs.

Training: The actual training of the CNNs comes into play when we receive the predictions and are able to calculate the cross entropy of them using the known labels of the input data. We then regularize the weights using L2 regularization and calculate the loss of the network [14]. We then reduce this loss via an Adam optimizer in order to locate the optimal weight and bias values and ultimately achieve a high accuracy with results which can be generalized [16].

The reason for implementing 10 CNNs of varying depths is to enable us to construct a voting system. Along with each CNN eventually providing an actual accuracy once data is passed into it, we will implement a voting system to exploit each networks accuracy and achieve the optimal accuracy by utilizing the model produced by each network.

III. Deep Transfer Learning for Traffic Sign Recognition

In order to develop transfer learning models to utilize data from other sources to improve the learning on a target dataset, we propose a few methods for utilizing a different dataset to provide beneficial knowledge and possibly outperform the TOLY implementation.

Similarly to TOLY classification, after preprocessing, we are able to input our data into TL models. Each of these models utilizes the same 10 CNN structure as the TOLY method,

however, each method modifies the network structure in order to implement TL in an elegant and innovative manner, and take advantage of previously learned data from other sources.

The transfer learning methods we propose are:

- Weight Initialization Transfer Learning (WITL)
- Feature Concatenation Transfer Learning (FCTL)
- Prediction Combination Transfer Learning (PCTL)

A. WITL: Weight Initialization Deep Transfer Learning

The first method of TL we implement is not a new method but has actually been one of the 4 recommended implementations of TL for the past few years [12]. This method takes into account the weight values of a network which have been pre-trained on auxiliary data and proposes initializing the weight values of the network training target data with the pre-trained values. It is typically recommended that this method be implemented when the target dataset is smaller than yet similar to the data in the auxiliary dataset.

We implement this method by training our auxiliary data using the same TOLY method as explained previously, however, as the network trains, we record the weight values in every convolutional layer as the network is trained, except for the final feature vector convolution. For example, in our 4-layer CNN we have 3 convolutional weight layers, excluding our feature vector convolution, therefore, by the end of training we should have 3 pre-trained weight values.

With the pre-trained weight values in our possession, we then focus on our target data. Before training the target data as we did in the TOLY method, we initialize the weight values in each convolution layer with the pre-trained values we extracted from our auxiliary data models. In order to prevent these weight values from changing during training, we hardcode these values into the weights of our target data networks. We leave our final feature convolution layer uninitialized so it can be trained and then train the network just like we did for TOLY classification.

The idea behind this method is to use the pre-trained weight values to teach our target data model what features may be of most importance based on what was learned from the auxiliary data set. This is why it is typical that the auxiliary data set is significantly larger than the target data set, because the large auxiliary data produces more detailed knowledge and is more beneficial to the smaller target data. Below are the corresponding algorithms used to implement the WITL method on auxiliary data or target data.

Algorithm 1 WITL - Auxiliary

Require: Data Matrix $\mathbf{X}_{ij} =< \mathbf{x}_{i}.\,, \mathbf{x}_{j}.\, >$.
Constructed **Weights**$_{aux}$, i.e., [**Weights**$_{aux}$] = [$\mathbf{W}_{.1}, \cdots, \mathbf{W}_{\cdots n}$]
Initialize $\mu = 0$ and $\sigma = 0.1$;
repeat
  $\mathbf{W} \leftarrow \frac{\phi(\mu,\sigma^2;\mathrm{x})}{\Phi(\mu,\sigma^2;\mathrm{b}) - \Phi(\mu,\sigma^2;\mathrm{a})}$;
  $\mathbf{B} \leftarrow$ One-dimensional array of zeros, size b;
  $\mathbf{C} \leftarrow$ 2-D convolution given 4-D input image, **X,** and truncated distribution, **W**;
  $\mathbf{R} \leftarrow \mathbf{R} + \frac{\mathbf{C}^2}{2}$;
  $\mathbf{C} \leftarrow$ max(0,**C**);
  Append **W** to **Weights**$_{aux}$;
  $\mathbf{C} \leftarrow$ maxpool(**C**);
until Converges;
$\mathbf{G}_{Final} = \mathbf{G} + \mathbf{G}_{s}$;
$\mathbf{F}_{Final} = \mathbf{F} + \mathbf{F}_{f}$;
Output: Array of weight vectors **Weights**$_{aux}$, containing converged weight values for each layer of the auxiliary data network, respectively.

B. FCTL: Feature Concatenation Deep Transfer Learning

Our next TL method focuses on utilizing the complete knowledge gained from both the auxiliary and target data set rather than treating one as more beneficial than the other. We aim to take advantage of the learned feature vectors from both sets of data and utilize the knowledge gained from both of them to potentially produce more accurate results compared to a TOLY classifier.

We begin by training our auxiliary data in the CNNs just as a normal TOLY classifier. However, once the network converges and training stops, we extract and save the learned feature vector from the model. We then train our target data in the same manner, but rather than build a single feature vector out of our convolutions and use only that for predicting, we concatenate our previously saved feature vector which was pre-trained on our auxiliary data. This provides us with a feature vector which is twice the size as the feature vector in our TOLY classification models. We then utilize this double-sized feature vector to make predictions on the target data during training. Figure 2 provides a visualization of training to predict target data utilizing this FCTL method.

Algorithm 2 WITL - Target

Require: Data Matrix $\mathbf{X}_{ij} =< \mathbf{x}_{i}.\,, \mathbf{x}_{j}.\, >$.
Weight array **Weights**$_{aux}$ = [$\mathbf{W}_{.1}, \cdots, \mathbf{W}_{\cdots n}$].
Initialize $\mu = 0$ and $\sigma = 0.1$;
repeat
  $\mathbf{W} \leftarrow$ **Weights**$_{aux_n}$;
  $\mathbf{B} \leftarrow$ One-dimensional array of zeros, size b;
  $\mathbf{C} \leftarrow$ 2-D convolution given 4-D input image, **X,** and truncated distribution, **W**;
  $\mathbf{R} \leftarrow \mathbf{R} + \frac{\mathbf{C}^2}{2}$;
  $\mathbf{C} \leftarrow$ max(0,**C**);
  $\mathbf{C} \leftarrow$ maxpool(**C**);
until Converges;
$\mathbf{G}_{Final} = \mathbf{G} + \mathbf{G}_{s}$;
$\mathbf{F}_{Final} = \mathbf{F} + \mathbf{F}_{f}$;
Output: Accuracy of predictions for target data traffic signs after initializing weights with auxiliary data values.

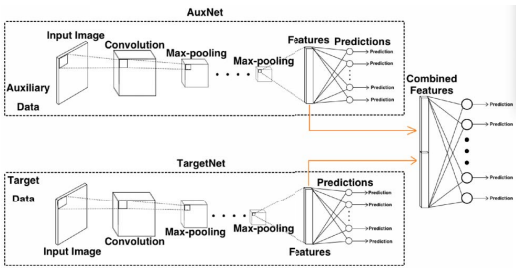

Fig. 2: Deep transfer learning via feature concatenation (using CNN). TargetNet denotes the target network, and AuxNet is the auxiliary network which is trained using auxiliary data (traffic signs). FCTL intends to train the target network (TargetNet) but combining feature output of AuxNet to form final feature set for classifying Target data traffic signs.

Now that we have a feature vector which is double the size and utilizing features learned from both the auxiliary data and the target data, we can perform testing on never before seen target data. We simply restore the feature vector we constructed in our target data models and apply it to a series of test traffic sign images to produce predictions. Figure 3 provides a visualization for our testing process utilizing the FCTL method, followed by the overall algorithm implementation.

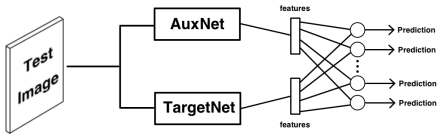

Fig. 3: Deep transfer learning model via feature concatenation (using CNN). TargetNet denotes the pre-trained model trained with target data, and AuxNet is the pre-trained model trained with auxiliary data. The feature vectors from both models are concatenated to make predictions on our never before seen traffic sign test images.

Algorithm 3 FCTL

Require: Auxiliary Data Matrix $\mathbf{X}_{ij} =< \mathbf{x}_i. , \mathbf{x}_j. >$.
Target Data Matrix $\mathbf{Y}_{ij} =< \mathbf{y}_i. , \mathbf{y}_j. >$.

Constructed **Weights**$_{aux}$, i.e., [**Weights**$_{aux}$] = [$\mathbf{W}._1, \cdots, \mathbf{W}..._n$]
Initialize $\mu = 0$, $\sigma = 0.1$, and **FC** $= 0$;
repeat
  $\mathbf{W} \leftarrow \frac{\phi(\mu,\sigma^2;x)}{\Phi(\mu,\sigma^2;b)-\Phi(\mu,\sigma^2;a)}$;
  B $\leftarrow$ One-dimensional array of zeros, size b;
  C $\leftarrow$ 2-D convolution given 4-D input image, **X**, and truncated distribution, **W**;
  $\mathbf{R} \leftarrow \mathbf{R} + \frac{\mathbf{C}^2}{2}$;
  C $\leftarrow$ max(0,**C**);
  C $\leftarrow$ maxpool(**C**);
  if FC $\neq 0$ **then**
    F $\leftarrow$ flatten(**C**);
    $\mathbf{F}_0 \leftarrow$ Concatenate(**F**, **FC**, 1);
    $\mathbf{F}_1 \leftarrow \frac{\phi(\mu,\sigma^2;x)}{\Phi(\mu,\sigma^2;b)-\Phi(\mu,\sigma^2;a)}$;
    $\mathbf{F}_{1b} \leftarrow$ One-dimensional array of zeros, size b;
    $\mathbf{R} \leftarrow \mathbf{R} + \frac{\mathbf{C}^2}{2}$;
    $\mathbf{L} \leftarrow \mathbf{F}_0 * \mathbf{F}_1 + \mathbf{F}_{1b}$;
  else
    F $\leftarrow$ flatten(**C**);
    $\mathbf{F}_1 \leftarrow \frac{\phi(\mu,\sigma^2;x)}{\Phi(\mu,\sigma^2;b)-\Phi(\mu,\sigma^2;a)}$;
    $\mathbf{F}_{1b} \leftarrow$ One-dimensional array of zeros, size b;
    $\mathbf{R} \leftarrow \mathbf{R} + \frac{\mathbf{C}^2}{2}$;
    $\mathbf{L} \leftarrow \mathbf{F}_0 * \mathbf{F}_1 + \mathbf{F}_{1b}$;
  end if
until Converges;
Output: Learned logits, **L**, regularizers, **R** and combined features, **FC**, respectively.

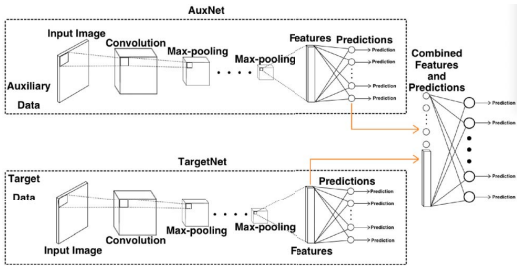

Fig. 4: Deep transfer learning model via prediction combination (using CNN). TargetNet denotes the pre-trained model trained with target data, and AuxNet is the pre-trained model trained with auxiliary data. PCTL intends to train the target network (TargetNet) by combining the predictions of AuxNet with the feature vector of TargetNet to form a final feature set for classifying Target data traffic signs.

C. PCTL: Prediction Combination Deep Transfer Learning

Our third method proposes the possibility that the predictions of a model trained on auxiliary traffic sign images contains more valuable information than the feature vector. This PCTL method is implemented in a similar manner to feature concatenation, only, rather than extracting the pre-trained feature vector from our network trained on auxiliary data, we extract the final predictions of this network and concatenate them to the feature vector of of the network training on target data.

Thus, we begin by training our CNNs on the auxiliary traffic sign data as normal and generate predictions. Once the network converges and finishes training, we save the predictions of the auxiliary training data. We then train the same CNNs on our target traffic sign data and rather than generate the single feature vector and use only it for prediction as in TOLY, we combine this vector with our resulting predictions from the training we performed on our auxiliary traffic sign data. The resulting feature vector is the size of the feature vector computed from flattening the previous convolution in the network, plus the learned predictions which is the size of the number of traffic sign labels we are predicting amongst. Figure 4 provides an illustration of this approach with both the auxiliary data predictions and the target data feature vector being used to make final predictions on the target data.

With this new feature vector containing features from the target training data and predictions from the auxiliary training data, we can perform testing on never before seen traffic sign test data. We accomplish this by restoring the pre-trained CNN models, extracting the combined feature vector, and using this to make predictions on the test data. Figure 5 provides a visualization of testing new data utilizing the knowledge gained from the predictions of our auxiliary traffic sign data and the feature vector of our target traffic sign data, followed by the overall PCTL algorithm implementation.

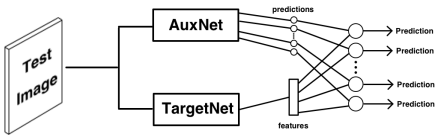

Fig. 5: An illustration of testing a deep transfer learning model via prediction combination (using CNN). TargetNet denotes the pre-trained model trained with target data, and AuxNet is the pre-trained model trained with auxiliary data. The feature vector from TargetNet is combined with the predictions from AuxNet to make final predictions on our never before seen traffic sign test images.

Algorithm 4 PCTL

Require: Auxiliary Data Matrix $\mathbf{X}_{ij} =< \mathbf{x}_i. , \mathbf{x}_j. >$.
Target Data Matrix $\mathbf{Y}_{ij} =< \mathbf{y}_i. , \mathbf{y}_j. >$.

Constructed **Weights**$_{aux}$, i.e., [**Weights**$_{aux}$] = [$\mathbf{W}._1, \cdots, \mathbf{W}..._n$]
Initialize $\mu = 0$, $\sigma = 0.1$, and **PC** $= 0$;
repeat
  $\mathbf{W} \leftarrow \frac{\phi(\mu,\sigma^2;x)}{\Phi(\mu,\sigma^2;b)-\Phi(\mu,\sigma^2;a)}$;
  B $\leftarrow$ One-dimensional array of zeros, size b;
  C $\leftarrow$ 2-D convolution given 4-D input image, **X**, and truncated distribution, **W**;
  $\mathbf{R} \leftarrow \mathbf{R} + \frac{\mathbf{C}^2}{2}$;
  C $\leftarrow$ max(0,**C**);
  C $\leftarrow$ maxpool(**C**);
  if PC $\neq 0$ **then**
    F $\leftarrow$ flatten(**C**);
    $\mathbf{F}_0 \leftarrow$ Concatenate(**F**, **PC**, 1);
    $\mathbf{F}_1 \leftarrow \frac{\phi(\mu,\sigma^2;x)}{\Phi(\mu,\sigma^2;b)-\Phi(\mu,\sigma^2;a)}$;
    $\mathbf{F}_{1b} \leftarrow$ One-dimensional array of zeros, size b;
    $\mathbf{R} \leftarrow \mathbf{R} + \frac{\mathbf{C}^2}{2}$;
    $\mathbf{L} \leftarrow \mathbf{F}_0 * \mathbf{F}_1 + \mathbf{F}_{1b}$;
  else
    F $\leftarrow$ flatten(**C**);
    $\mathbf{F}_1 \leftarrow \frac{\phi(\mu,\sigma^2;x)}{\Phi(\mu,\sigma^2;b)-\Phi(\mu,\sigma^2;a)}$;
    $\mathbf{F}_{1b} \leftarrow$ One-dimensional array of zeros, size b;
    $\mathbf{R} \leftarrow \mathbf{R} + \frac{\mathbf{C}^2}{2}$;
    $\mathbf{L} \leftarrow \mathbf{F}_0 * \mathbf{F}_1 + \mathbf{F}_{1b}$;
  end if
until Converges;
Output: Learned logits, **L** and regularizers, **R**, respectively.

IV. Experiments

A. Benchmark Datasets

With the goal of transfer learning being to improve learning of a target task utilizing data from other sources, we analyzed three different types of traffic sign data.

The initial dataset consists of 2,603 images of US traffic signs which were locally gathered from a vehicle dashboard camera. Since TSR is a multiclass classification problem, this set of images contains 43 different classes of traffic signs, each represented by a different amount of images in the set. Each image was taken during the mid-day hours so that there was adequate lighting as well as consistent features to be learned across the dataset. Additionally, we categorized each image into one of 43 categories depending on the traffic sign the image contained. We then constructed a file system to house these images and labeled them with a numerical value between 0 and 43 corresponding to what traffic sign they represented. These labels become the output predictions of our experiment networks for TSR.

The second set of images come from the GTSRB dataset and provides us with a much larger and different set of traffic sign images [17]. This dataset contains 39,209 images of 43

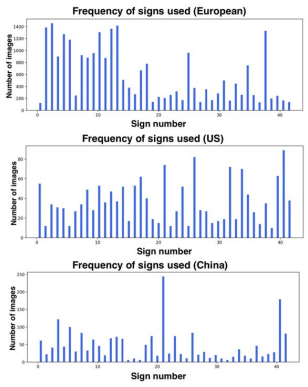

Fig. 6: Class distributions of the three traffic sign datasets.

Fig. 7: Examples of traffic sign images from each dataset.

different types of European traffic signs.

The third dataset consists of Chinese traffic sign images gathered from the National Nature Science Foundation of China (NSFC) [18]. In order to maintain consistency with the other datasets, we selected 43 types of Chinese traffic signs to focus on from this dataset which provided us with 3,394 images in total, each numerically labeled based on the traffic sign in the image. Figure 6 provides a visualization of how often each sign appears in each dataset.

The fact that these datasets differ greatly in their amount of images, yet the signs in the images differ only slightly in granular details like shape and sign graphics, as emphasized in Figure 7, provides a solid foundation for TL experimentation. We propose that the data should blend well and each set could provide beneficial knowledge to the other when utilized in an effective TL method.

B. Baseline Methods

With our data gathered and labeled, we begin the process of implementing it into our 4 previously explained methods for TSR. The initial step before inputting the sign images into our CNNs is to preprocess each image according to the technique described in section 3. This provides us with a randomly mixed set of 64x64x1 grayscale images which are slightly clearer and increased in contrast. We believe this should make it easier for our models to learn unique features among the 43 different traffic signs in each dataset.

1) TOLY

As explained in section 3, The TOLY method does not utilize transfer learning and instead relies on a DL architecture of 10 CNNs of varying depth to train the model on only the target data and ultimately provide predictions for this same target data.

US-Net

We first perform our TOLY method on US dataset. This results in 10 models capable of recognizing US traffic signs to varying degrees of accuracy. Each model builds a feature vector containing learned features which it deems significant for recognizing US traffic signs. Additionally, each model ends with building a vector of predictions for the traffic signs it was trained on. We call this set of models USNet to represent the fact that it is trained on US traffic sign data.

Euro-Net

We then perform the same TOLY training on the EU data GTSRB. Similar to USNet, each of the 10 models we obtain builds a feature vector for recognizing European traffic signs and a prediction vector based on the training data. We also label this set of models as EuroNet, based on the data on which it was trained.

China-Net

Lastly, we perform the very same TOLY training on the Chinese data gathered from NSFC. As before, each of the 10 models we obtain builds a feature vector – this time for recognizing Chinese traffic signs – and a prediction vector based on the training data. We also label this set of models as ChinaNet, based on the data on which it was trained.

Naive TL

For the sake of providing further motivation for the possible benefit of TL implementations we also performed a data switch test in which we utilize one of our pre-trained model sets to recognize test data from the other dataset. In our case, we tested the individual ability of EuroNet to recognize US sign data and USNet to recognize European sign data.

2) WITL

To implement WITL, we begin by obtaining all pre-trained weights of our 10 trained EuroNet models as well as separately obtaining our 10 trained ChinaNet models. We then take these weight values and hardcode them into the corresponding convolutional weight layers of our target data's CNN networks so they do not change upon training. However, we leave our final, fully-connected convolutional weight layer uninitialized so it is able to be trained. With these target data CNNs now initialized to the pre-trained weight values, we then train the networks twice just as if we were utilizing the TOLY method, once for using EuroNet weights and once for using ChinaNet weights.

3) FCTL

We implement this method on USNet, EuroNet, and ChinaNet by first training whichever one we will extract the learned feature vector from. Once this training is complete, we train our target dataset like normal. However, rather than only build a single feature vector out of previous convolutions in the network, we add to this vector by concatenating the previously learned feature vector from the other dataset to it and we use this larger feature vector to build a series of classifiers and make predictions on our target data.

4) PCTL

The PCTL method is implemented by extracting the final 43 predictions from the models which trained on our auxiliary data. We then train our target data networks with the target data like normal, but instead of building a feature vector from only the previous convolution, we also add the 43 predictions we extracted from our pre-trained models. This adds 43 values to our original target feature vector and we use this vector to make predictions on our target data.

C. Experimental Settings

1) Tools used

Our experiment was performed utilizing the TensorFlow library to carry out network training, testing, and transfer learning [19]. While there are many other substantial libraries for implementing deep learning strategies and experiments, we chose TensorFlow due to it's ability to let us control every element in our networks which proved essential for implementing TL. While the learning curve with it is fairly steep, TensorFlow allowed us to perform crucial operations on our network weights, features, and any other necessary elements. All experiment was performed on a Late 2013 MacBook Pro housing a 2.6 GHz Intel Core i7 processor, 16 GB of memory, and two GPUs, an NVIDIA GeForce GT 750M with 2048 MB of memory and an Intel Iris Pro with 1536 MB of memory.

2) Data preparation

The USNet data was split into training and testing sets during the gathering process. Of these 2,603 images, 1,986 were gathered exclusively for training. This set of 1,986 images was then cross validated randomly so that 80% (1,588) were used as our training set and 20% (398) were used for a validation set. The other 617 images were gathered exclusively for testing so that there was no chance of the network containing a very similar image in both the test set and training set. This made the final US dataset ratio approximately 76% training/validation and 24% testing. Each sign image is represented by a prediction label ranging from 0 to 43 and each of these labels is mapped to a string containing the actual name of the sign in a CSV file. Eventually, we will reference this file to display the sign name on an actual image.

The EuroNet data was gathered as a single large dataset of 39,209 European traffic sign images. We performed random cross-validation so that 80% of these images, amounting to 31,367, were randomly allocated to a training set and 20% (7,842 images) was allocated to the test set. Then, we performed random cross-validation again on the 31,367 images so 20% of this training set (6,274 images) was allocated to a validation set.

The final dataset of Chinese traffic signs (ChinaNet) consisted of 3,394 traffic sign images. As with the other two datasets, we performed random cross-validation two separate times which resulted in 60% (2,043 images) of the images being allocated to a training set, 15% (511 images) allocated to a validation set, and the remaining 25% (840 images) belonging to the test set.

V. Experimental Results and Analysis

In order to observe the results of our experiments and the effect of our TL algorithms, we perform a series of statistical analyses to gauge the significance of our TL results beginning with assessing the accuracy. The first accuracy we discuss is the result of the previously explained naive TL method. This is the effect of utilizing the models which are pre-trained on auxiliary data to recognize the target data.

We can compute that if the sign images for each dataset were represented evenly across the US dataset and European dataset, which they are not, then the odds of randomly selecting the correct sign label for recognition, out of 43 potential options, is about 2.32%. Thus, since we know that the amount of sign images in the dataset varies for each type of sign, we can presume that this probability will most likely be lower on average since some signs are highly represented while others appear less often in the data. The results of utilizing these datasets to recognize one another without TL can be observed in Table I as fairly similar to random selection.

Obviously, this method provides extremely low accuracy, even in the case where our auxiliary data is much larger than our target data. Thus, an elegant and more strategic implementation of TL could be of far greater value.

European traffic sign prediction using network trained from US signs	
Network Architecture	Classification Accuracy (43 classes)
4-Layer	1.17%
6-Layer	1.08%
9-Layer	2.32%
9-1x1-Layer	1.52%
11-Layer	2.30%
12-Layer	3.47%
13-Layer	2.01%
14-Layer	1.40%
15-Layer	2.40%
16-Layer	1.40%
Vote of above Networks	1.82%

US traffic sign prediction using network trained from European signs	
Network Architecture	Classification Accuracy (43 classes)
4-layer	2.59%
6-layer	0.65%
9-layer	0.00%
9-1x1-layer	0.16%
11-layer	1.62%
12-layer	1.94%
13-layer	5.19%
14-layer	0.32%
15-layer	1.94%
16-layer	1.46%
Voting of above Networks	1.30%

TABLE I: The classification accuracy of applying target data to networks pre-trained from auxiliary data. For each row, auxiliary data (US traffic signs) are used to trained a deep neural network. The target data (European traffic signs) are directly applied to the trained networks. The results confirm that directly applying target data to pre-trained auxiliary networks results in very poor accuracy.

A. Statistics

Next, we analyze the results of each previously proposed TSR method, beginning by reporting each method's accuracy across each of the 10 CNNs in the system. Based on these results, we compute the mean and standard deviation to begin performing a T-test in order to test whether any accuracy improvements are significantly higher than the TOLY model. Our T-test begins with declaring a null and alternate hypothesis:

$$H_0 : \bar{x_1} \not> \bar{x_2} \tag{3}$$

$$H_1 : \bar{x_1} > \bar{x_2} \tag{4}$$

Where $\bar{x_1}$ is the mean accuracy of our TL method and $\bar{x_2}$ is the mean accuracy of our TOLY method.

We then label alpha level α as 0.05 and compute that degrees of freedom df is 20 since each datasets will have 11 accuracy values, including voting, and we are comparing two sets at a time. Therefore, since we are testing if our TL method has significantly greater accuracy than our TOLY method, we want to perform a one-tail T-test with a T-critical value as 1.725. We will use the mean and standard deviation values to calculate our t-statistic using the following formula:

$$t = \frac{\bar{x_1} - \bar{x_2}}{\frac{S}{\sqrt{n}}} \tag{5}$$

where n is the dataset size and S is the TL standard deviation.

B. Results: Using EuroNet to predict USNet

We begin performing our experiment by utilizing USNet data as target data. Table II shows the accuracy of each method.

From the above data, we can observe that each of the 3 TL models outperform the TOLY implementation in at least some if not all of the architectures. However, while this could signify a potential benefit, it is important that we decipher whether these results are statistically significant or just the outcome of random chance and unlikely to be able to be replicated.

In order to assess the significance of the result differences, we first compute the mean accuracy and standard deviation of

Classification accuracy of each model utilizing USNet as target data				
Depth	TOLY	WITL	FCTL	PCTL
4-Layer	86.94%	91.80%	83.14%	88.01%
6-Layer	80.50%	75.27%	81.52%	89.95%
9-Layer	79.11%	81.98%	81.85%	87.36%
9-1x1-Layer	71.73%	84.85%	88.65%	77.63%
11-Layer	81.39%	77.31%	84.28%	85.90%
12-Layer	81.74%	80.97%	80.39%	86.55%
13-Layer	80.75%	83.53%	87.36%	88.82%
14-Layer	81.32%	81.41%	78.44%	90.60%
15-Layer	82.39%	85.98%	87.03%	87.36%
16-Layer	80.96%	83.50%	83.14%	86.39%
Voting of above networks	90.92%	93.03%	92.87%	93.84%

TABLE II: Classification accuracy of TSR methods utilizing USNet data as target dataset and EuroNet data as auxiliary dataset. TOLY represents the target only model trained using only USNet as the target data and tested with a USNet test set of traffic sign images. WITL uses the pre-trained weight values of EuroNet to initialize the weights of USNet. FCTL concatenate pre-trained features of both USNet and EuroNet to make predictions on USNet traffic sign data. PCTL combines the pre-trained EuroNet predictions with the pre-trained USNet feature vector to make predictions on the USNet target data.

Mean and standard deviation of models using USNet as target data				
	TOLY	WITL	FCTL	PCTL
Mean	81.66%	83.78%	84.42%	87.49%
Std. Dev.	4.4936	5.1083	3.9893	3.8134

TABLE III: Mean accuracy and standard deviations of the classification accuracy of each TSR method listed in Table II. These methods use USNet traffic signs as the target data, and EuroNet traffic signs as the auxiliary data.

each TSR method. Note that our baseline is our TOLY model accuracy so the objective of each TL model is to improve on this accuracy. Table III shows the mean accuracy and standard deviation for each recognition method.

Finally, we want to test whether the TL results are significantly different enough from our TOLY method to claim that they offered a beneficial improvement. To do this, we perform a T-test between our TOLY method results and each of our TL method results.

Since our t-critical value is 1.725, we can observe that for the weight initialization method, the t-statistic falls within this range and therefore, we can accept the null hypothesis. This means that this TL method's mean accuracy most likely was only greater than the TOLY method by chance and is not significantly greater. The t-stat values of both the feature concatenation and prediction combination methods fall outside the t-critical value range and into the rejection range, therefore we can reject the null hypothesis and accept that these mean accuracies are significantly greater than the accuracy of our TOLY method.

C. Results: Using USNet to predict ChinaNet

The next situation we test with our proposed methods is one in which two datasets are similar in data size. We begin by utilizing knowledge gained from our USNet data, which

T-test results with USNet data as target data		
WITL	FCTL	PCTL
1.3764	2.2946	5.0704

TABLE IV: T-statistics of our 3 TL methods compared with our TOLY classification method where US signs are our target data and EU signs are our auxiliary data

Classification accuracy of each model utilizing USNet as auxiliary data				
Depth	TOLY	WITL	FCTL	PCTL
4-layer	85.83%	70.31%	87.11%	83.90%
6-layer	82.98%	60.78%	85.12%	80.80%
9-layer	84.17%	48.21%	80.43%	77.69%
9-1x1-layer	78.21%	55.30%	84.18%	73.22%
11-layer	84.05%	53.19%	81.78%	73.67%
12-layer	84.17%	52.89%	88.15%	82.53%
13-layer	86.19%	59.80%	89.87%	82.13%
14-layer	80.60%	67.31%	91.24%	87.54%
15-layer	85.00%	52.41%	92.22%	80.80%
16-layer	81.90%	58.91%	90.77%	89.78%
Voting	88.68%	61.31%	92.32%	84.52%

TABLE V: The classification accuracy of the proposed TSR methods. TOLY represents models trained using only the ChinaNet sample as the target data and tested with a sampled ChinaNet test set of traffic sign images. WITL uses the pre-trained weight values of USNet to initialize the weights of ChinaNet. FCTL concatenates the pre-trained features of both ChinaNet and USNet to make predictions on ChinaNet traffic sign data. PCTL combines the pre-trained USNet predictions with the pre-trained ChinaNet feature vector to make predictions on the ChinaNet target data.

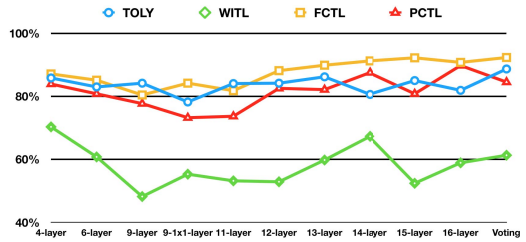

Fig. 8: A visualization of the classification accuracy for each traffic sign recognition method listed in Table V.

contains 2,603 images of US traffic signs, to predict traffic signs in our ChinaNet dataset, which contains 3,394 images of Chinese traffic signs. Table V below shows the accuracy of each network predicting Chinese traffic signs from USNet data using the various methods and figure 8 shows a visualization of the accuracies in a line graph.

We can observe from these results that the only method which provided an improved mean accuracy is the FCTL method. Therefore, we utilize a T-test to observe the significance of the improvement and observe a t-critical value of 2.9749, thus providing support that our proposed FCTL can provide statistically significant improved results over TOLY methods when the datatsets are similar in size.

D. Results: Using ChinaNet to predict USNet

In order to provide further robustness to our experiment regarding similarly sized datasets, we perform the same test as before, however, rather than use USNet as auxiliary data, we utilize ChinaNet as auxiliary data and USNet as our target data. Table VI shows the accuracy of each network performing this experiment as well as a line graph in Figure 9.

E. Results: Using USNet to predict EuroNet

Lastly, we provide results of the same tests, however, we utilize USNet as our auxiliary data and EuroNet as our target data. The goal here is to observe the effect our TL methods have when the target dataset is much smaller in size than the auxiliary dataset. Figure 10 reports the accuracy of each method.

From the above data we can observe that the results of our proposed TL methods are less accurate than our TOLY classification implementation. We attribute these results to the

Classification accuracy of each model utilizing ChinaNet as auxiliary data				
Depth	TOLY	WITL	FCTL	PCTL
4-layer	86.94%	79.20%	89.22%	81.11%
6-layer	80.50%	73.89%	85.90%	78.56%
9-layer	79.11%	74.80%	81.79%	80.02%
9-1x1-layer	71.73%	76.22%	82.32%	71.31%
11-layer	81.39%	73.10%	85.59%	73.33%
12-layer	81.74%	75.35%	81.68%	74.59%
13-layer	80.75%	79.40%	89.98%	77.64%
14-layer	81.32%	80.81%	92.41%	80.79%
15-layer	82.39%	78.93%	92.35%	78.30%
16-layer	80.96%	71.57%	89.75%	76.80%
Voting	90.92%	83.19%	92.97%	84.33%

TABLE VI: The classification accuracy of the proposed TSR methods utilizing the USNet data as target dataset and ChinaNet data as auxiliary dataset. TOLY represents models trained using only the USNet sample as the target data and tested with a sampled USNet test set of traffic sign images. WITL uses the pre-trained weight values of ChinaNet to initialize the weights of USNet. FCTL concatenates pre-trained features of both ChinaNet and USNet to make predictions on USNet traffic sign data. PCTL combines the pre-trained ChinaNet predictions with the pre-trained USNet feature vector to make predictions on the USNet target data.

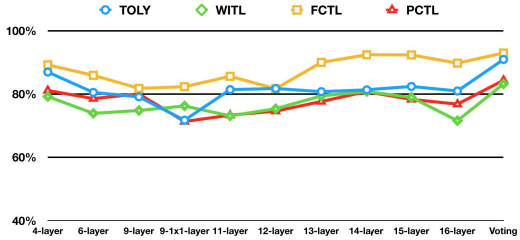

Fig. 9: A visualization of the classification accuracy for each traffic sign recognition method listed in Table VI.

fact that our auxiliary data is roughly 3,000 images and our target data consists of roughly 40,000 images. Due to this massive data imbalance, the networks appear to be disregarding any pre-trained values such as the weights in our WITL implementation, feature vector in FCTL, and predictions in PCTL. Since these values were trained on such a smaller set of data, they do not provide as much beneficial information as the large amount of target data.

VI. Conclusion

In this paper, we proposed to use deep transfer learning for traffic sign recognition. We proposed two methods, FCTL and PCTL, which demonstrated significant performance gain compared to simple approaches. Our results also provide evidence that as our models increased in the number of layers, the accuracy increased as well. Specifically, when auxiliary dataset was significantly larger than target dataset, both methods provided significantly improved results. However, when the auxiliary and target data both contain approximately the same amount of data, the proposed FCTL method provided improved accuracy over TOLY classification. Furthermore, we observed that when the target dataset is very large and the auxiliary dataset is very small, our proposed methods did not provide improved accuracy results over the TOLY method. It is also important to note that this research was focused on static images of standard traffic signs rather than dynamic signs or traffic signals such as lights. Future research can experiment with these variables as well as dataset sizes and number of network layers.

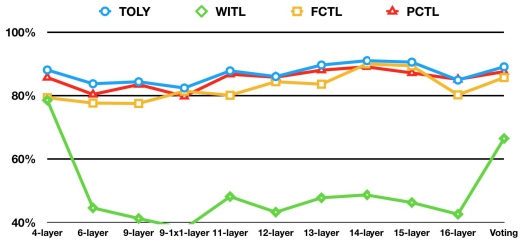

Fig. 10: The classification accuracy of each TSR method utilizing EuroNet data as target dataset and USNet data as auxiliary dataset.

References

[1] Y. Fang, D. Shen, Z. Xiao, X. Xin, and Y. Zeng, "Traffic sign recognition using kernel extreme learning machines with deep perceptual features," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1647–1653, November 2016.

[2] N. Keren, T. Miyoshi, H. Nakayasu, P. Patterson, and Y. Seya, "Measurement of visual attention and useful field of view during driving tasks using a driving simulator," *Proceedings of the 2007 Mid-Continent Transportation Research Symposium*, August 2007.

[3] M. Fang, J. Yin, X. Zhu, and C. Zhang, "Trgraph: Cross-network transfer learning via common signature subgraphs," *IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(9)*, pp. 2536–2549, 2015.

[4] A. Karpathy and J. Johnson, "Transfer learning," [Online] http://cs231n.github.io/transfer-learning/, 2016.

[5] R. Timofte, K. Zimmermann, and L. V. Gool, "Multi-view traffic sign detection recognition and 3d localisation," *Machine Vision and Applications*, pp. 1–15, December 2011.

[6] F. Larsson and M. Felsberg, "Using fourier descriptors and spatial models for traffic sign recognition," *Proc. Image Anal.*, pp. 238–249, 2011.

[7] C. Grigorescu and N. Petkov, "Distance sets for shape filters and shape recognition," *IEEE Trans. Image Process*, vol. 12, no. 10, pp. 1274–1286, October 2003.

[8] R. Belaroussi, P. Foucher, J. Tarel, B. Soheilian, P. Charbonnier, and N. Paparoditis, "Road sign detection in images: A case study," *Proc. ICPR*, pp. 484–488, 2010.

[9] D. Ciresan, U. Meier, and J. Schmidhuber, "Transfer learning for latin and chinese characters with deep neural networks," *Neural Networks (IJCNN)*, July 2012.

[10] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014.

[11] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013.

[12] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *CoRR*, vol. abs/1411.1792, 2014.

[13] B. Zhao, B. Huang, and Y. Zhong, "Transfer learning with fully pretrained deep convolution networks for land-use classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 9, pp. 1436–1440, September 2017.

[14] L. Rosasco and T. Poggio, *A Regularization Tour of Machine Learning*. MIT Press, 2015, no. MIT-9.520 Lectures Notes.

[15] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, Aug 2007.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[17] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: A multi-class classification competition," *Proc. IJCNN*, pp. 1453–1460, 2011.

[18] L. Huang, "Chinese traffic sign database."

[19] G. Inc., "Tensorflow," http://www.tensorflow.org, 2016.