Final Project – Transfer Learning Tutorial

**IDS-705 Principles of Machine Learning**

*Yuan Feng, Sebastián Soriano Pérez, Vishaal Venkatesh,*

*Abhiraj Vinnakota, Roderick Whang*

**Abstract** – This tutorial explains the concepts and motivations behind transfer learning and seeks to provide an example of the technique used in practice. When tackling a machine learning problem, we may face a situation in which we do not have enough training data in the domain we are interested in. However, there is sufficient data available in a related domain. Transfer learning takes advantage of the available data in a related domain to train a model for our target domain. Transfer learning has been shown to improve performance and reduce the costs of acquiring more labeled data. This tutorial will focus on applying transfer learning to deep neural networks for image classification problems, using a feature extraction (embedding) approach. We will first explain the methods and formal concepts behind transfer learning, and then provide an example to measure and understand its performance.

1. **Introduction**

[Provide a description concept/algorithm, motivate your reader as to why he/she should care about this question. How has the technique been used in practice and what are some motivating examples of its use?]

1. **Background**

[This section should cite problems that have been previously addressed that relate to your work, and the key takeaways of the studies that explored that work. The idea here is to place the problem you’re working on in context and to let the reader know that you’re not working in a knowledge vacuum. For finding relevant literature, a good starting point is Google Scholar.]

1. **Methods**

In order to understand transfer learning, we must define two important concepts first: *domain* and *task*. In a machine learning, the **domain** consists of the feature space in the dataset, with features or predictors (), and its marginal probability distribution [1]. The marginal probability distribution is the probability distribution of each of the variables with no reference to the values of the rest of the variables. A **task** consists of the label space and an objective predictive function , which can be used to predict the class label and corresponds to the conditional probability distribution learned from the training data (where ) [1]. For a graphical representation see Figure 4.1.

Transfer learning is done by first training a machine learning model with training data having a source domain and a source task . For a binary classification problem, the training dataset would consist of pairs of values, where , the feature vector is , and . It is assumed that there is sufficient training data available to train the machine learning model, and this data is usually balanced. Then, transfer learning utilizes the information learned from this model and applies it to train new model with data having a specific target domain and a target task . Transfer learning aims to improve the learning of the target predictive function using the knowledge acquired by training the source model, when or [1]. Therefore, it is possible to try to estimate the target conditional probability distribution by using the information found through the training of the source model. See Figure 4.2.

The transfer learning techniques can be applied to different scenarios, depending on whether it is the target and source domains, or tasks that differ. The domains differ when the source and target feature spaces are different () or when the source and target marginal probability distributions are different (). Likewise, the tasks differ when the source and target label spaces are different (), when the conditional probability distributions are different (), or both. There is a wide range of transfer learning techniques that could be applied in any of these situations.

**Inductive transfer learning** is applied when the source and target tasks are different, regardless if their respective domains are the same or not. **Transductive transfer learning** is used when the source and target tasks are the same, but their respective domains are different. Finally, **unsupervised transfer learning** is applied to problems where the target task is different and focuses on unsupervised learning problems such as clustering or dimensionality reduction. We will focus on and provide examples of inductive transfer learning techniques from now on, particularly when there is plenty of labeled data in the source domain and the focus is to improve performance on a smaller set of labeled data available on the target domain.

Transfer learning is particularly popular for training neural networks for image classification, especially for convolutional neural networks. When we need to train a classification model but there is limited amount of labeled data available, we can use the information obtained from training a different model with more reliable data in a related domain. Image classification problems is an instance where the application of inductive transfer learning methods (when the source and target marginal probability distributions differ) is particularly useful.

A neural network for image classification consists of an input layer where each neuron corresponds to one of the features (where ), a varying number of hidden layers, and an output layer where each neuron corresponds to one of the labels of . The general idea behind how a neural network “learns” is that each layer after the input layer captures slightly more complex features in the images. The first layers’ neurons activate when there are specific edges in certain regions of the images, and subsequent layers start detecting more complex patterns and shapes. The final layer receives as an input more easily identifiable and complex features, so that it is easier to correctly detect which features are found on each label of .

A common approach to applying transfer learning for computer vision is to utilize a previously trained model on data with the same feature space but different marginal probability distribution, when the labeled data is sufficient. Then, the final layers of the pre-trained neural network are “removed”, and the remaining layers and their weights are fixed and used as feature extraction models for the new data in the target domain and task. The final layers of the model are then retrained again on the target data. This process is shown in Figure 4.3. This approach is known as **feature extraction** or **embedding**. This method improves the training time and model performance. In the following section we will apply the feature extraction technique as an example.

1. **Examples of the Technique in Practice**

To put this technique to practice we first trained a ResNet-18 convolutional neural network from scratch using the *hymenoptera\_data* dataset [2]. The ResNet-18 architecture consists of 18 convolutional layers [more description missing] [3].

We first trained the ResNet-18 network, which will be referred to as Model 1 from now on, with a balanced training dataset of 246 labeled images of ants (124 images) or bees (122 images). See Figure 4.2 for some samples of the images in the dataset. While the exact amount of training data required to train a model depends on many factors such as number of features, number of classes, or type of model, a rule of thumb indicates that a minimum of 1000 images is usually required to train a computer vision model [4]. We did not expect the model to perform very well because of this. The *hymenoptera\_data* dataset also includes a validation set of 155 labeled images (71 of ants and 84 of bees). We calculated the model’s loss on both the training and validation sets during each epoch, and we used the validation set to measure the model’s performance on unseen data.

We set the hyperparameters of Model 1 to 30 epochs, a starting learning rate of 0.001 and a learning rate decay factor of 0.1 every 7 epochs. We also used cross entropy loss as the loss function during backpropagation and the model was trained in batches of 4 images. Model 1 took 30 minutes and 13 seconds to train; it achieved a best accuracy of 0.7039 on the validation set during epoch 24 and an AUC value of 0.7882. The confusion matrix for the model’s performance on the validation set with a threshold value of 0.5 is shown in Figure 4.3 along with a sample of five images and their predicted labels. While Model 1’s performance was considerably better than chance, its training time is too long, and we will prove that its performance can be improved through transfer learning.

We then proceeded to train a new model, Model 2, using inductive transfer learning. The PyTorch library for Python provides a ResNet-18 model pre-trained on a subset of the ImageNet dataset, consisting on 1000 different classes and more than one million images [5]. The model weights for every layer are obtained through this method and are then initialized into Model 2. The pre-trained model weights were trained to recognize patterns on an enormous variety of images of many different classes. The last layer before the output of the ResNet-18 model contains information on several general features found across all 1000 classes on the ImageNet dataset. When the weights of the pre-trained model are frozen, they can be used to pre-process any image and output activations on each of these general features, in order to best describe which ones are contained in the image. This is the feature extraction or embedding model we discussed in the previous section.

We now train Model 2 with our *hymenoptera\_data* dataset and our frozen weights from the ImageNet model. The result of the forward propagation of every batch of images going through the model training is the embedding of each of them into the features found by the ImageNet-trained model. This will make it easier for Model 2 to identify which features are only found in pictures of bees and which ones are found in pictures of ants. Model 2 was trained using the same hyperparameters as Model 1. It took 13 minutes and 38 seconds to train; it achieved a best accuracy of 0.9474 on the validation set during epoch 29 and an AUC value of 0.9684. The confusion matrix for the model’s performance on the validation set with a threshold value of 0.5 is shown in Figure 4.4 along with a sample of five images and their predicted labels.

Model 2 represents a reduction of 54.88% in training time, an improvement of 34.59% in accuracy, and a 22.86% increase in AUC. See Figure 4.4 for a comparison of Model 1 and Model 2’s ROC and Precision-Recall curves. It is now evident that the use of transfer learning technique can improve training time and performance very significantly. The one disadvantage there exists about transfer learning is that there has to be an existing pretrained model on a related domain available for you to be able to use. For computer vision and image classification tasks this is not an issue as there is a wide range of available pre-trained models publicly available. However, if one were to undergo the training process from scratch on a new domain, the collection of enough labeled data and training of a new model would be expensive and time-consuming.

Ants vs. Bees

Imagen que contiene foto, interior, diferente, tabla

Descripción generada automáticamente

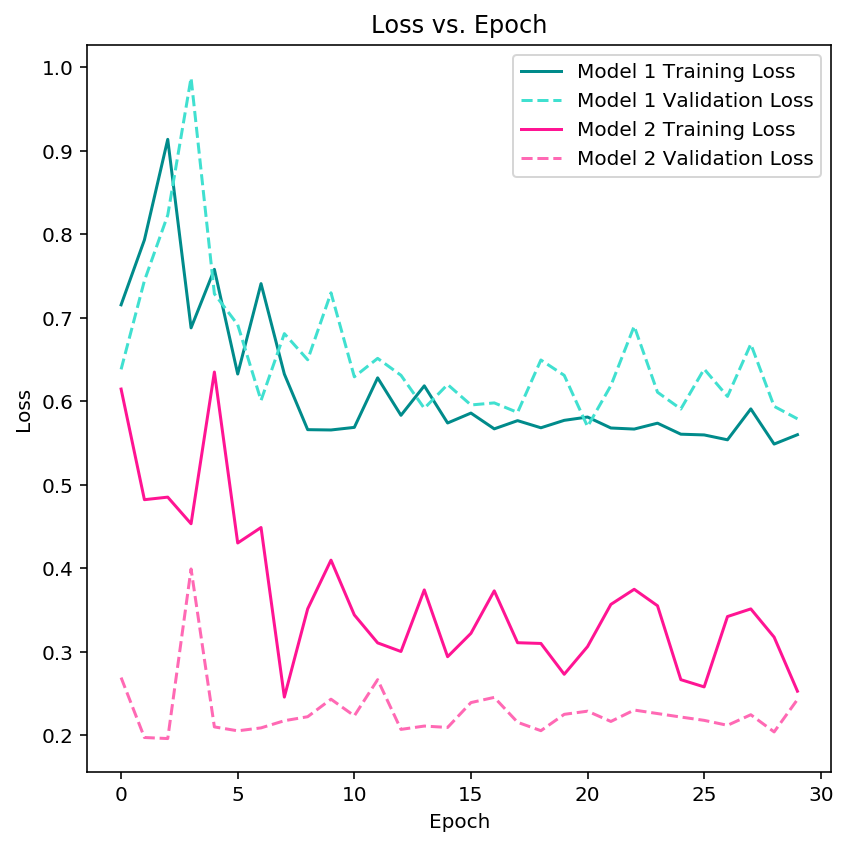


Imagen que contiene texto, mapa

Descripción generada automáticamenteImagen que contiene texto, mapa

Descripción generada automáticamente

1. **Summary**

[Summary of the method, its uses, as well as its strengths and weaknesses as compared to other similar techniques.]

1. **Roles**

**Yuan Feng**: Worked on Abstract, Introduction and Summary. Collaborated building non-quantum models.

**Sebastián Soriano Pérez**: Researched and authored Methods and Examples of techniques in practice sections (non-quantum parts). Implemented non-quantum models. Co-authored Background section.

**Vishaal Venkatesh**:

**Abhiraj Vinnakota**: Researched and authored Background section.

**Roderick Whang**: Worked on Abstract, Introduction and Summary. Built figures and plots. Collaborated with research and Background section.

1. **References**

[1] Pan, Sinno Jialin, and Qiang Yang. “A Survey on Transfer Learning.” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, Oct. 2010, pp. 1345–59. DOI.org (Crossref), doi:10.1109/TKDE.2009.191.