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 often do not have enough training data in the domain we are interested in, but there is sufficient data available in a related domain. Transfer learning takes advantage of the available data 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 with classical computing and a classical-quantum computing hybrid 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

In the supervised learning context, deep neural networks have become more and more accurate in recent years. Convolutional neural networks using a residual learning network are particularly good at finding the optimal model parameters while improving accuracy [1]. However, it is not possible to apply these deep models into a different situation or phenomena other than what they were trained and designed for. It is important to have this flexibility in real-world scenarios where we rarely encounter data structured in the same way as our training data. Transfer learning allows us to transfer some of the information learned from an ideal situation into a different and related situation for which our data is more limited.

In this paper we will define a common transfer learning technique for computer vision and image classification problem: inductive transfer learning using feature extraction or embedding. By using a model previously trained on a robust training dataset, we can take advantage of its optimal parameters as a way to extract features for new data, reducing training time and improving performance considerably. Finally, we will discuss one particular cutting-edge approach to this problem using quantum computing for a hybrid transfer learning method. In this approach, we replace a neural network's output layer for a quantum computing network that we then train for our target domain and task [7].

2. Background

Pan, Sinno Jialin, and Qiang Yang have provided the following framework to understanding transfer learning [2]. In a machine learning, the **domain** $\mathcal{D} = \{\mathcal{X}, P(X)\}$ consists of the feature space $\mathcal{X}$ in the dataset, with p features or predictors ($X = \{x_1, \dots, x_p\} \in \mathcal{X}$), and its marginal probability distribution $P(X)$. The marginal probability distribution $P(X)$ is the probability

distribution of each of the variables $x_1, \ldots, x_p$ with no reference to the values of the rest of the variables. A **task** $\mathcal{T} = \{\mathcal{Y}, f\}$ consists of the label space $\mathcal{Y}$ and an objective predictive function f, which can be used to predict the class label $f(x)$ and corresponds to the conditional probability distribution $P(Y|X)$ learned from the training data $\{X, Y\}$ (where $Y \in \mathcal{Y}$) [2].

We first train a machine learning model with data having a source domain $\mathcal{D}_S$ and a source task $\mathcal{T}_S$. For a binary classification problem, the training dataset would consist of n pairs of $\{\boldsymbol{x}_i, y_i\}$ values, where $i = 1, \ldots, n$, the feature vector is $\boldsymbol{x}_i = [x_{i,1} \quad \cdots \quad x_{i,p}] \in X$, and $y_i \in Y$. It is assumed that there is sufficient training data available to train the machine learning model, and this data is usually balanced. Then, we use the information learned from this model and apply it to train new model with data having a related target domain $\mathcal{D}_T$ and task $\mathcal{T}_T$. Transfer learning aims to improve the learning of the target predictive function f_T using the knowledge previously acquired, when $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. Therefore, it is possible to estimate the target conditional probability distribution $P(Y_T|X_T)$ by using the information found through the training of the source model (See Figure 2.2). Some authors have proposed ways to quantify the relationships between different tasks, such as affinity matrices [5].

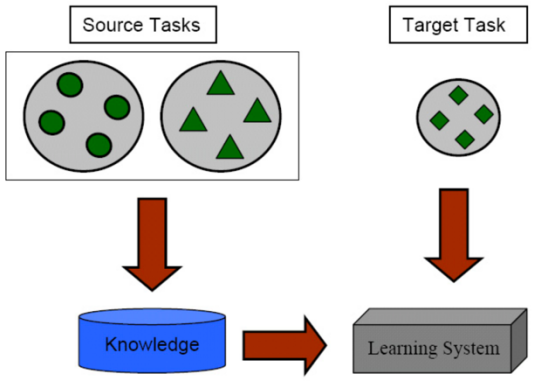

Figure 2.1:
Knowledge transfer in Transfer Learning [2].

Inductive transfer learning is applied when the source and target tasks are different, regardless if their respective domains are the same or not (when the source and target label spaces are different, $\mathcal{Y}_S \neq \mathcal{Y}_T$, or when the conditional probability distributions are different, $P(Y_S|X_S) \neq P(Y_T|X_T)$). Transductive transfer learning is used when the source and target tasks are the same, but their respective domains are different (when the source and target feature spaces are different, $\mathcal{X}_S \neq \mathcal{X}_T$, or when the source and target marginal probability distributions are different, $P(X_S) \neq P(X_T)$). 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 for a particular inductive transfer learning technique.

Inductive transfer learning techniques allow us to be very flexible when selecting the information that we want to transfer from one model to another. One cutting-edge approach is

the implementation of quantum computing techniques, creating a hybrid classical-to-quantum (CQ) neural network. CQ transfer allows to pre-train a model with a classical deep neural network and successively manipulate the extracted features after removing the last layers with a variational quantum circuit [6]. Despite the early stages of quantum computing, CQ transfer learning approaches can compete with classical methods, which is very promising for the future of this area. We will now explain the basics of quantum computing in order to understand how the quantum circuits work.

The fundamental unit of quantum information is the quantum bit or **qubit**, analogous to the bit in a classical computer. A qubit exists in a linear combination of the zero-state ($|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$) and the one-state ($|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$), represented by the equation $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Here, α and β are the probability amplitudes of the zero- and one- states, respectively, and their squares are the probabilities for each of their corresponding states to occur after being measured (thus $\alpha^2 + \beta^2 = 1$). In other words, the probabilistic nature of qubits only holds until they are measured, and then the qubit assumes the value of either $|0\rangle$ or $|1\rangle$, depending on its probability amplitudes. Quantum computers work by applying quantum gates (analogous to logic gates in classical computing) on the qubits. States with non-zero amplitudes after the application of the gate have a chance of being the actual value when measured, determined by the squares of their resulting probability amplitudes. For instance: after applying a common quantum gate, the Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$, to the zero-state qubit (1), we obtain a 0.5 chance of it taking a value of $|0\rangle$ and 0.5 of $|1\rangle$ (2).

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\end{bmatrix} \quad (1)$$

$$\alpha^2 = \frac{1}{2} \text{ and } \beta^2 = \frac{1}{2} \quad (2)$$

There are a handful of problems where quantum computers offer an exponential speedup relative to their classical counterparts. The hope is that more such problems will be identified in the future with advancements in quantum information science. The application of a CQ transfer model using quantum circuits to image classification problems is not expected to provide a quantum speedup. However, we wish to implement this cutting-edge approach with the goal of evaluating its performance on top of ResNet-18 and compare it to its classical counterpart.

3. Methods

Image classification 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 x_j (where $j = 1, \ldots, p$), a varying number of hidden layers, and an output layer where each neuron corresponds to one of the labels of Y. On a neural network, each layer after the input layer capture 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 Y.

A common approach to applying transfer learning for computer vision is to utilize a previously trained model on data with the same feature space and transfer its trained parameters (**parameter transfer**). 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, since they are intuitively more representative of the features found in the data they were originally trained for [3]. This process is shown in Figure 3.1 below. 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.

To put this technique to practice we will use the ResNet-18 convolutional neural network architecture by training it from scratch (Model 1) using the *hymenoptera_ data* dataset [4]. The ResNet-18 architecture consists of 18 convolutional layers using 3×3 kernels and shortcut connections with residual mappings [1]. Each layer has a number of weights that connect the node's activations to the following layer's nodes. We have provided a more in-depth description of the ResNet architecture in our Kaggle report. We will apply a classical transfer learning approach (Model 2) and a CQ hybrid transfer learning approach using a 4-qubit dressed quantum circuit (Model 3) on the last layer to compare the performances of all three models.

We will use a model pre-trained on the 1000-class ImageNet dataset to transfer the weights for the first 17 layers. The model's final layer will consist of 512 input features and 2 final outputs (ants or bees). Figure 3.1 shows this transfer learning process, where A is the pre-trained model on the ImageNet dataset, A' is this same trained network minus its final layer, B is a second layer or group of layers that will be retrained on the target domain and for the target task. Our examples consist of the following settings:

- $\mathcal{D}_S$: 1000-class ImageNet dataset [7].
- $\mathcal{T}_S$: 1000 labels classification.
- $\mathcal{D}_T$: 246 labeled images of ants (124 images) and bees (122 images) [4].
- $\mathcal{T}_T$: 2 labels classification (ants or bees)

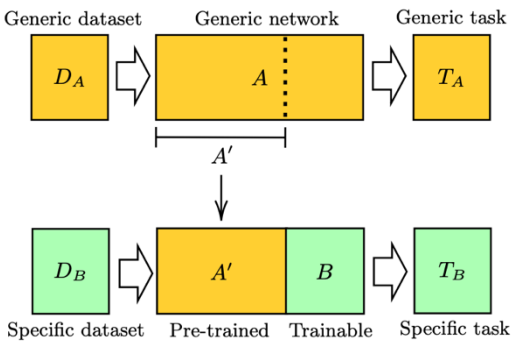

Figure 3.1:
Inductive Transfer Learning with a feature extraction or embedding approach [6].

4. Examples of the Technique in Practice

For Model 1, we trained the ResNet-18 network on domain $\mathcal{D}_T$ and task $\mathcal{T}_T$ from scratch. See Figure 4.1 for some samples of the images in the dataset. We did not expect the model to perform very well because the training data is way below 1000 images [4]. The *hymenoptera_ data* dataset also includes a validation set of 153 labeled images (70 of ants and 83 of bees). We obtained 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 (See Figure 4.5 below for a comparison with Models 2 and 3's loss per epoch).

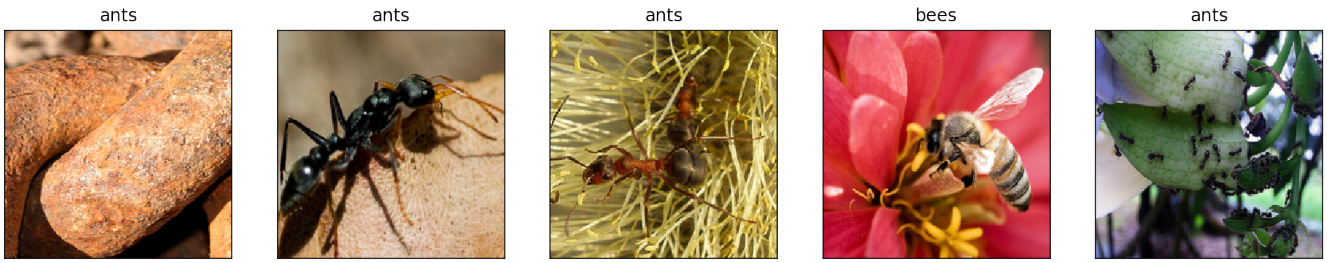

Figure 4.1:
Sample images from the hymenoptera_data *dataset.*

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 achieved a best accuracy of 0.7255 on the validation set during epoch 11 and an AUC value of 0.7458. The confusion matrix for the model's performance on the validation set with a threshold value of 0.5 is shown in Figure 4.2 along with a sample of five images and their predicted labels.

Model 1		Predicted Class	
		Ants	Bees
Actual Class	Ants	54	16
	Bees	26	57

Figure 4.2:
Confusion matrix for Model 1.

For Model 2, A' is a ResNet-18 model pre-trained on a subset of the ImageNet dataset $\mathcal{D}_S$ minus its 18^{th} layer. When the weights of the pre-trained model are frozen, they can be used to pre-process any image as a feature extraction method and output activations each of the 512 general features in the last hidden layer, in order to best describe which ones are contained in the image. We train the last layer B with our *hymenoptera_ data* dataset $\mathcal{D}_T$ for our two classes

$\mathcal{T}_T$. Model 2 was trained using the same hyperparameters as Model 1. It achieved a best accuracy of 0.9542 on the validation set during epoch 19 and an AUC value of 0.9793. 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. Model 2 represents a reduction of 56.99% in training time, an improvement of 31.52% in accuracy, and a 31.31% increase in AUC over Model 1.

Model 2		Predicted Class	
		Ants	Bees
Actual Class	Ants	67	3
	Bees	4	79

Figure 4.3:
Confusion matrix for Model 2.

For Model 3, A' is the same as Model 2 and is still used as a feature extraction method. B now is the 4-qubit dressed quantum circuit trained with our *hymenoptera_ data* dataset $\mathcal{D}_T$ for our two classes $\mathcal{T}_T$. Model 3 was trained using the same hyperparameters as Model 1. It achieved a best accuracy of 0.9608 on the validation set during epoch 25 and an AUC value of 0.9757. 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. Model 3 represents an improvement of 32.43% in accuracy, and a 30.83% increase in AUC over Model 1. However, Model 3 took considerably longer to train when compared to Model 1.

Model 3		Predicted Class	
		Ants	Bees
Actual Class	Ants	65	5
	Bees	1	82

Figure 4.4:
Confusion matrix for Model 3.

See Figure 4.5 for a comparison of the ROC and Precision-Recall curves for Models 1, 2, and 3. It is now evident that the use of transfer learning technique can improve training time and performance very significantly. The loss functions decrease with both the classical and CQ approaches, while both AUC and AP improve by a considerable amount over the model built from scratch. 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.

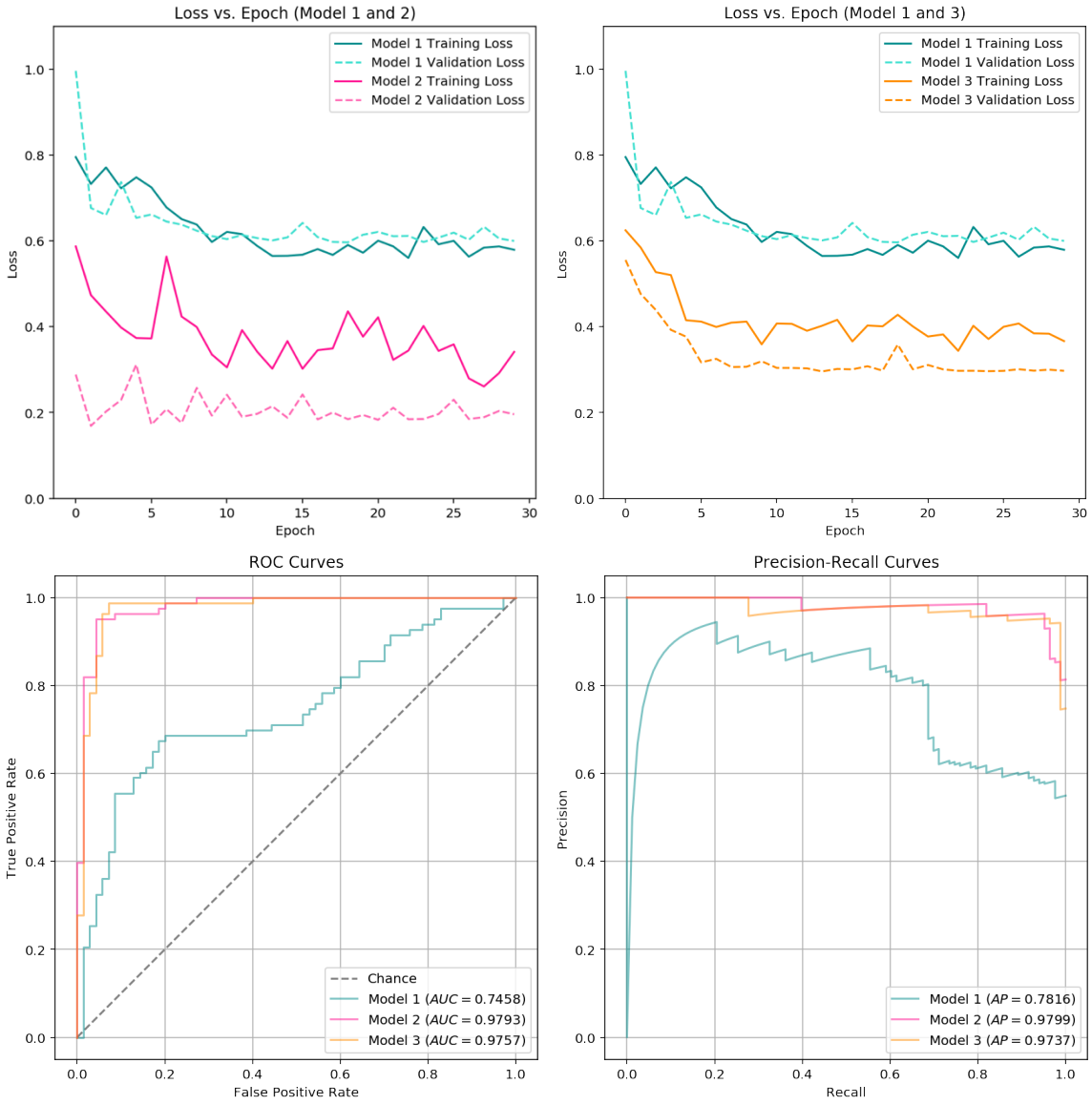

Figure 4.5:

Top left: Loss vs. Epoch for models 1 and 2. Top right: Loss vs. Epoch for models 1 and 3. Bottom left: ROC curves for models 1, 2, and 3. Bottom right: Precision-Recall curves for models 1, 2, and 3.

5. Summary

Transfer learning techniques can improve significantly on similar models built from scratch. Training time can be reduced by one half with classical parameter transfer methods, and performance in terms of AUC, AP and accuracy can also be improved significantly. These results can be attained with a minimum amount of training data as long as there is a pre-existing model trained in a similar domain. Both the classical and the CQ hybrid approach accomplished very similar performance in terms of accuracy and AUC.

6. Roles

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

Sebastián Soriano Pérez: Researched and authored Background and Methods (classical methods). Implemented classical models.

Vishaal Venkatesh: Researched and authored Background and Methods (CQ hybrid model). Implemented CQ hybrid model.

Abhiraj Vinnakota: Researched and co-authored Background section. Organization and tutoring regarding ResNet-18 model implementations.

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

7. References

[1] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." *ArXiv:1512.03385* [Cs], Dec. 2015. arXiv.org, http://arxiv.org/abs/1512.03385.

[2] 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.

[3] Razavian, Ali Sharif, et al. "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition." *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2014, pp. 512–19. DOI.org (Crossref), doi:10.1109/CVPRW.2014.131.

[4] "Hymenoptera_data." *Kaggle*, https://www.kaggle.com/ajayrana/hymenoptera-data.

[5] Zamir, Amir, et al. "Taskonomy: Disentangling Task Transfer Learning." *ArXiv:1804.08328 [Cs]*, Apr. 2018. arXiv.org, http://arxiv.org/abs/1804.08328.

[6] Mari, Andrea, et al. "Transfer Learning in Hybrid Classical-Quantum Neural Networks." *ArXiv:1912.08278 [Quant-Ph, Stat]*, Dec. 2019. arXiv.org, http://arxiv.org/abs/1912.08278.

[7] Sasank Chilamkurthy, *PyTorch transfer learning tutorial*. https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html. Accessed: 2020-04-13.