Transfer Learning in a Jigsaw Puzzle-like Image Similarity Task

Nhung Hoang and Ellen Liu

December 18, 2017

Abstract

Transfer learning takes advantage of previously learned models to expedite the training of related tasks. We applied the concept of transfer learning to a pair of neural networks. The first network uses convolution and pooling layers to extract features from images. The second network uses the features discovered by the first network to classify whether pairs of puzzle piece images derived from the original images fit together. To assist in this difficult task, the second network is also trained on two auxiliary tasks: if the puzzle pairs shapes fit together and if the two puzzles are of the same image. The results of our network demonstrate that transfer learning can be a promising tool for image classification.

1 Introduction

Image classification, the task of assigning images certain labels or categories is a skill that comes naturally to humans but is a difficult skill for a machine to learn. In recent years, deep learning and data mining models focusing on feature extraction, feature transformation, and pattern analysis have been explored and extensively studied as successful solutions to the difficulties that come with general classification [1][2][3][4].

Convolutional Neural Networks (CNNs) in particular have been held as the dominant architecture for image recognition and classification tasks. Previous CNN work has focused on improving recognition rates using modifications to network architecture like parameter tuning, or using groups of convolutional neural networks and averaging their outputs [5].

One main drawback of machine learning technologies is the lack of generalization. The assumption made in traditional supervised learning problems is that a model intended for a specific task and domain should be provided with training data for that task and that domain. For example, in a model that has been trained on detecting pedestrians in day-time images, it would be difficult to then apply that model to a dataset with pedestrians in night-time images because the model is unable to generalize to the new domain of the night-time background [6].

1.1 Transfer Learning

To combat the generalization problem, researchers came up with the transfer learning technique. This is a skill humans naturally apply to new problems: knowing how to play one instrument will make it somewhat easier to learn another, knowing how to solve mathematical equations prepare students for applying themselves to learning physics. In a general sense, transfer learning happens when knowledge in one context enhances the learning and performance in a new and separate context.

By using previously trained CNNs, transfer learning can help decrease the time and effort needed to generate the amount of training data adequate for a network to learn the most important features of images. In this paper, we approach transfer learning from a slightly different perspective. We aim to exploit and build on the transfer learning technique using convolutional neural networks for detecting image similarity. In particular, we focus on an aspect of a jigsaw puzzle task: detecting whether or not inputs are puzzle pieces of the same image or different images. The human approach to solving a jigsaw puzzle includes distinguishing pieces by color and patterns, and eventually, the shape or cutout of the puzzle piece edges. We train a network to classify an existing dataset of images based on different features. We then use the features found to train a second network on detecting the similarity between image pieces. We hypothesize that a CNN will be able to differentiate between images using their colors and shapes.

1.2 Related Work

Our experiments were motivated by recent studies in the applications of transfer learning to complex image classification problems. Researchers have explored the possibilities of transferring feature sets learned from feeding one dataset into a network to map onto new datasets that are similar to, but not the same as the first.

A study on language and character recognition focuses on transfer learning with deep neural networks on different handwritten languages. The researchers concluded that it is possible to transfer learning between recognizing uppercase letters, Chinese characters, and Latin letters [7]. Their experiments resulted in successful use of transfer learning on image classification with an optimized deep neural network.

CNNs require a very large number of image samples to be used in a training dataset, which poses a problem when there is limited data for a task at hand. In previous studies [8][9], work on transfer learning has tried to overcome the deficit of training samples for some categories. The proposed solution is to train a neural network on existing large datasets. In one such study, researchers demonstrated the success of initially training a convolutional neural network on the ImageNet dataset, a large visual database that has millions of image samples [8]. They pre-trained a network on a subset of the ImageNet data to classify centered objects in an image. Their initial network was then successfully used to classify objects from a different dataset, outperforming regular neural network performance on the new dataset.

Another approach to transfer learning attempts to address the aforementioned difficulty of obtaining an adequate training set where too few labeled examples are available. A solution to this problem is to use pseudo tasks that are created from data without supervision in order to automate the process of initially training the network for later tasks [9]. Pseudo-task construction involves sampling a random patch from one image and applying it as a filter that operates on other images.

The pseudo-tasks are tasks automatically generated in order to create more tasks for a network to learn and gain knowledge from in preparation for a new, unrelated task.

Our work aims to expand on the simpler transfer learning in an image classification task by first training a network to learn important features from an image dataset, then manipulating the shapes of the same images. We will take the idea of transfer learning in visual recognition and apply it to detecting image similarity between two manipulated pieces of the same or different images.

2 Methods

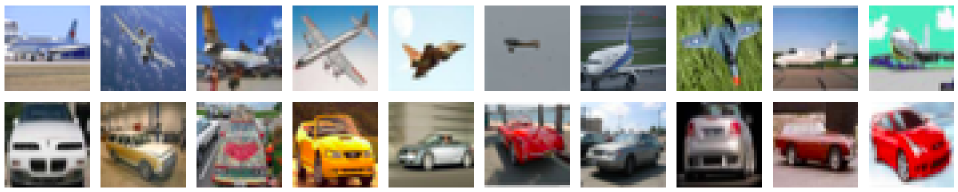

Figure 1: A subset of random images from the CIFAR-10 dataset in two different classes.

For our experiments, we use images from the CIFAR-10 dataset [10]. It consists of 60,000 color images of dimensions 32x32x3. The images are evenly split into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck; each with 6,000 images (Figure 1). The images are publicly available for use in the machine learning and computer vision community.

2.1 Synthetic Puzzle Pieces

For our second dataset to be used in the transfer learning task, we processed CIFAR images into puzzle-like pieces. The 5 puzzle piece shapes we chose to create ranged from the simplest pattern to the most challenging. Puzzle piece A was the simplest, with just a straight edge down the middle of the image or images. Puzzle piece E was the most complex, with an uneven array of cutouts along its edge (Figure 2). We used an artificially bright green color to block out the unwanted parts of the images because there is a low chance of that color showing up naturally in any image in the dataset.

Within each of the ten classes, we split the images according to each of the five puzzle piece shapes. This resulted in 12,000 puzzle piece cutouts in each class, with 300 images of each of the puzzle piece shapes. The puzzle pieces were combined into pairs with 30,000 negative sample pairs and 30,000 positive sample pairs. The positive pairs are made up of two pieces of the same image. The negative pairs mismatch in three different ways: (1) mismatched puzzle piece shape, (2) mismatched object, (3) mismatched piece and mismatched object. This splits the negative pairs evenly into 10,000 pairs for each negative subset.

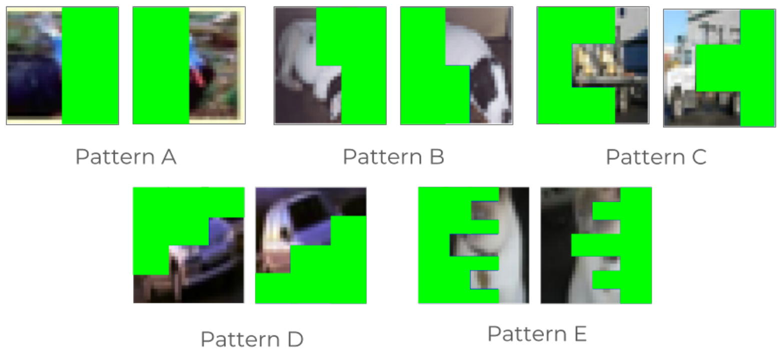

Figure 2: Different puzzle piece shapes used to manipulate the images from the CIFAR dataset.

2.2 Puzzle Pair Labels

The label of an image pair is an array of arrays: $\Big[[\mathrm{A}]\ [\mathrm{B}]\ [\mathrm{C}]\ [\mathrm{D}]\Big]$. The four arrays, A, B, C, and D, consist of multiple labels to indicate whether or not an image pair possesses a certain property (1.0) or not (0.0).

$$\text{A: } \Big[\text{Pattern of first image}\Big] \text{ - [Pattern A, Pattern B, Pattern C, Pattern D, Pattern E]}$$
$$\text{B: } \Big[\text{Pattern of second image}\Big] \text{ - [Pattern A, Pattern B, Pattern C, Pattern D, Pattern E]}$$
$$\text{C: } \Big[\text{Same or different image}\Big] \text{ - [Same Image, Different Image]}$$
$$\text{D: } \Big[\text{Fit of two images}\Big] \text{ - [Fit, No Fit]}$$

The pattern of the first image in a pair is always the left version of the pattern; the second image's pattern is always the right version. Arrays A and B hold the output for the respective auxiliary tasks of puzzle shape identification. Array C is the output of the image similarity task, which seeks to identify if the partial images from a pair of puzzle pieces is from the same image. For the fit task, we consider two puzzle pieces to fit together if they contain the same image and they have corresponding pattern shapes. Array D outputs the second network's prediction for this property.

A positive image pair with pattern D would have the output:

$$[\quad [0.0, 0.0, 0.0, 1.0, 0.0], \quad [0.0, 0.0, 0.0, 1.0, 0.0], \quad [1.0, 0.0], \quad [1.0, 0.0] \quad] \tag{1}$$

A negative image pair, where the first puzzle piece is of pattern A, the second puzzle piece is of pattern B, and the pieces show different images, would have the output:

$$[\quad [1.0, 0.0, 0.0, 0.0, 0.0], \quad [0.0, 1.0, 0.0, 0.0, 0.0], \quad [0.0, 1.0], \quad [0.0, 1.0] \quad] \tag{2}$$

2.3 Network Architecture

2.3.1 CIFAR-10 Feature-Extraction Network

Layer	Type	Parameters
0	Input	32x32x3
1	Convolutional	32 filters, 3x3 kernel size, ”relu” activation
2	Convolutional	32 filters, 3x3 kernel size, ”relu” activation
3	Max Pooling	2x2 size, 0.25 dropout
4	Convolutional	64 filters, 3x3 kernel size, ”relu” activation
5	Convolutional	64 filters, 3x3 kernel size, ”relu” activation
6	Max Pooling	2x2 size, 0.25 dropout
7	Flatten	N/A
8	Dense	512 nodes, ”relu” activation, 0.5 dropout
9	Dense (class output)	10 nodes, ”softmax” activation

Table 1: Feature extraction network architecture and parameter settings.

We used the original CIFAR-10 images to train a simple convolutional neural network with a succession of convolutional and max pooling layers (Table 1). The last layer uses a softmax activation function so that each neuron's output is able to be interpreted as belonging to a certain class. During training, we used a batch size of 32 input image pairs for 50 epochs. We saved the weights learned in this network and propagated our own manipulated image pairs through to get each image's feature representation.

2.3.2 Puzzle Similarity Detection Network

Layer	Type (Name)	Parameters
0	Input (puzzle1)	feature vector of shape (512,)
1	Input (puzzle2)	feature vectors of shape (512,)
1	Dense (dense1)	4 nodes, ”relu” activation
2	Dense (dense2)	4 nodes, ”relu” activation
3	Concatenate (concat3)	8 nodes
4	Dense (dense3)	8 nodes, ”relu” activation
4	Dense (dense4)	8 nodes, ”relu” activation
5	Dense (pattern1_out)	5 nodes, ”softmax” activation
6	Dense (pattern2_out)	5 nodes, ”softmax” activation
7	Dense (image_out)	2 nodes, ”softmax” activation
8	Dense (fit_out)	2 nodes, ”softmax” activation

Table 2: Puzzle piece network architecture and parameter settings.

The resulting feature representations from the CIFAR-10 feature extraction network were then used as inputs to our new convolutional network (Table 2, Figure 3). Each feature vector is fed through a dense layer. The two dense layers merge together into one, which then goes into 2 more 8-node

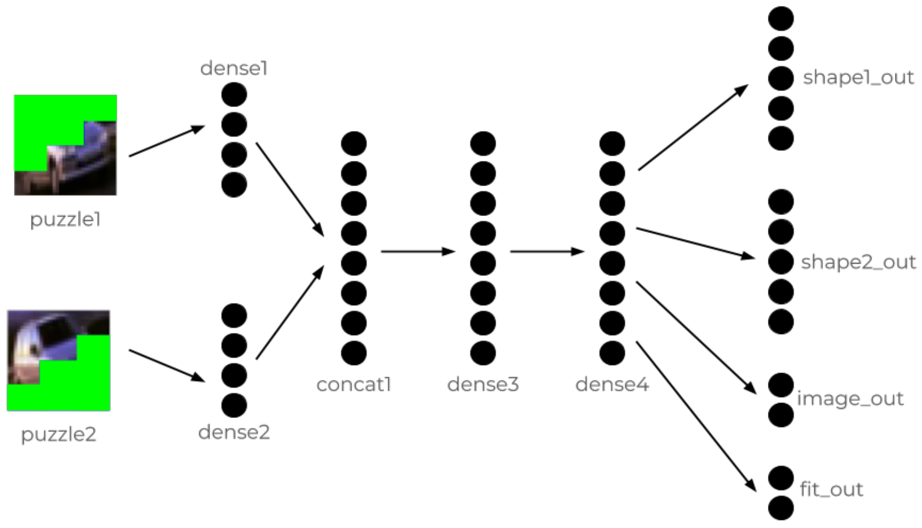

Figure 3: Architecture of the puzzle piece network.

dense layers. The final one of those dense layers results in a collection of the four output arrays which indicate the shape of the first image, the shape of the second image, whether the image pieces are the same, and whether the images fit together.

2.4 System of Combined Networks

The general framework for our network system is visualized in Figure 4. In summary, we propagate the puzzle pieces of one input pair separately through the CIFAR-10 Feature-Extraction Network. We take the two resulting, highest-level feature vectors and propagate them through the 2-input Puzzle Similarity Detection Network. This network returns four outputs describing what the network has predicted about the puzzle piece pair.

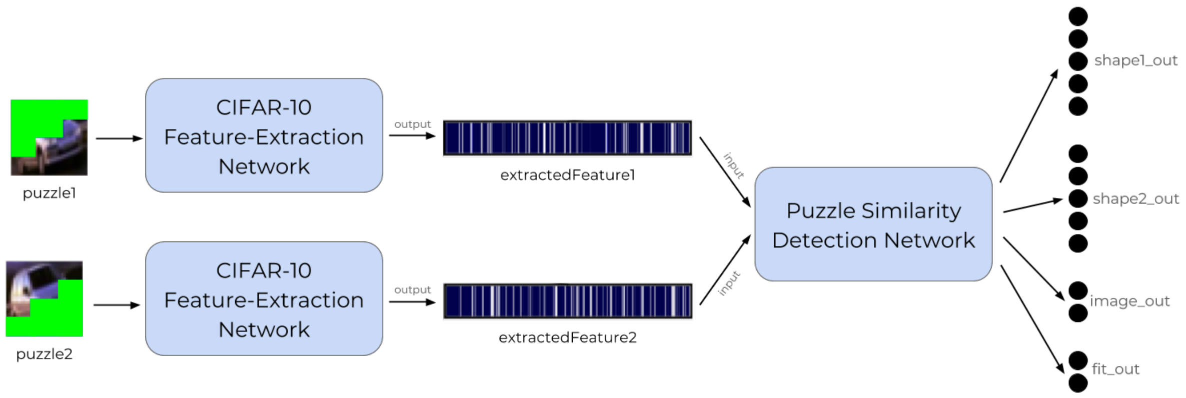

Figure 4: Framework for propagating inputs through the system of networks.

2.5 Experimentation

We trained the CIFAR-10 Feature-Extraction Network using the original CIFAR-10 image dataset with an 80/20 train/test split. We trained the Puzzle Similarity Detection Network using the set of paired, highest-level extracted-feature vectors also with an 80/20 train/test split. Then, we ran five independent experiments on the trained system of networks where we propagated a test set of feature vector pairs through the second network and recorded the accuracy.

3 Results

3.1 Feature-Extraction

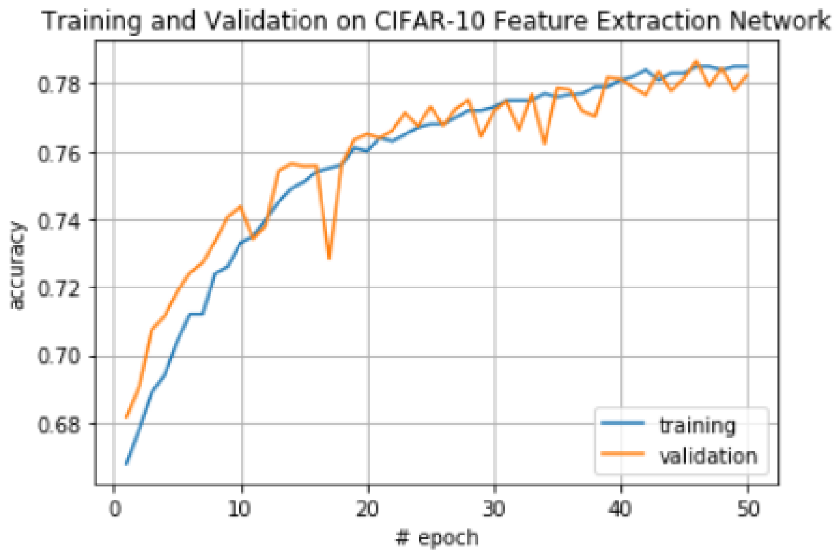

Figure 5: Results of training the feature extraction network on the original CIFAR-10 dataset. The orange line represents the validation set, and the blue line represents the training set.

The CIFAR-10 Feature-Extraction network reached a test accuracy of 78.25% for the original dataset. In creating the synthetic puzzle pieces, we did not keep track of the original labels of the images we were altering. Thus, we do not have evidence for how well this network was able to identify the objects in the puzzle pieces given the partial obscurity (the artificial green block coloring). However, we can use the results of the Puzzle Similarity Detection network to make a hypothesis for how well this first network was able to make predictions. As shown in Figure 5, the network (using the parameters we set) has not yet reached a maximum point of performance, which we define as the point just before the model overfits to the training data. Despite this, the model has a respectable accuracy, although future work can involve further fine-tuning the model's parameters to achieve better accuracy.

3.2 Puzzle Similarity Detection

As shown in Table 3, the network was able to correctly predict whether or not two puzzle pieces fit together with an average test accuracy of 79.14%. For the auxiliary tasks, the network's averaged test accuracies were: 87.78% for identifying the puzzle shape of the left piece, 83.75% for identifying the right piece's puzzle shape, and 66.82% for recognizing if the pair's images are the same or not.

Run	**Shape 1**	**Shape 2**	**Image**	**Fit**
1	86.3	83.9	66.8	79.0
2	89.5	84.3	66.6	79.6
3	89.3	83.8	66.7	79.4
4	89.3	84.0	67.2	78.9
5	84.4	82.7	66.8	78.8
Avg (%)	87.78	83.75	66.82	79.14

Table 3: Accuracy (%) of network in predicting each output over 5 runs of 50 epochs.

4 Discussion

From the third auxiliary task, puzzle image similarity, we speculate that the feature extraction network would have had a more difficult time with labeling the synthetic puzzle images. All of the puzzle patterns obscured approximately half of the original image, so it is not surprising that the system of networks had a hard time predicting the images from two puzzle pieces are from the same image or not. From the results, it seems that the shapes of the puzzle patterns were more useful than the puzzle image for the task of determining similarity between two puzzle pieces. There is no significant difference between the accuracies of those two tasks, meaning that identifying the left puzzle shape was as important as identifying the right puzzle shape. This makes intuitive sense, as humans tend to have the same mentality. Overall, the puzzle similarity detection network was able to identify if two puzzle pieces fit together or not using the outputs of the feature extraction network's last hidden layer. The performance of the image similarity auxiliary task was most likely inhibited by the CIFAR-10 Feature-Extraction network's difficulty with learning features for partially obscured images. The shape auxiliary tasks had high performance scores that most likely aided the main fit task.

4.1 Future Work

This system of networks is able to achieve a respectable level of accuracy for the main fit task. To further analyze what the model is extracting from the puzzle pair for this task, we would like to add more auxiliary tasks that highlight different properties of the puzzle pieces. For example, we would like to observe the distribution of colors in the puzzle pieces. We hypothesize that this property will be more beneficial than the image similarity task to the second network. Additionally, we would like to extend this puzzle similarity detection task to become a general puzzle task for a model, involving more than two puzzle pieces. In this domain, we would train the model to determine where each puzzle piece fits relative to the other pieces. Lastly, we want to compare the performance of this system of networks, involving transfer learning, to the performance of one network with image inputs, which is the traditional approach. From this experiment, we want to observe how helpful transfer learning is to the puzzle similarity detection task, in terms of computational efficiency and prediction accuracy.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[2] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, 2003.

[3] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16, 2003.

[4] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22, 2010.

[5] Dan Claudiu Ciresan, Ueli Meier, and Luca Maria Gambardella. Convolutional neural network committees for handwritten character classification. *Document Analysis and Recognition (ICDAR)*, 2011.

[6] Sebastian Ruder. Transfer learning - machine learning's next frontier, 2017.

[7] Dan C. Ciresan, Ueli Meier, and Jurgen Schmidhuber. Transfer learning for latin and chinese characters with deep neural networks. *WCCI 2012 IEEE World Congress on Computational Intelligence*, 2012.

[8] Maxime Oquab, Ivan Laptec Leon Bottou, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[9] Amr Ahmed, Kai Yu, Wei Xu, Yihong Gong, and Eric Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. *Computer Vision ECCV 2008: 10th European Conference on Computer Vision*, 2008.

[10] Alex Krizhevsky, Vinod Nair, and Geofrey Hinton. Learning multiple layers of features from tiny images. 2009.