Knowledge Transfer in Deep Convolutional Neural Nets

Steven Gutstein, Olac Fuentes and Eric Freudenthal

Computer Science Department
University of Texas at El Paso
El Paso, Texas, 79968, U.S.A.
{sgutstein,ofuentes,efreudenthal}@utep.edu

Knowledge transfer is widely held to be a primary mechanism that enables humans to quickly learn new complex concepts when given only small training sets. In this paper, we apply knowledge transfer to deep convolutional neural nets, which we argue are particularly well suited for knowledge transfer. Our initial results demonstrate that components of a trained deep convolutional neural net can constructively transfer information to another such net. Furthermore, this transfer is completed in such a way that one can envision creating a net that could learn new concepts throughout its lifetime.

The experiments we performed involved training a Deep Convolutional Neural Net (DCNN) on a large training set containing 20 different classes of handwritten characters from the NIST Special Database 19. This net was then used as a foundation for training a new net on a set of 20 different character classes from the NIST Special Database 19. The new net would keep the bottom layers of the old net (i.e. those nearest to the input) and only allow the top layers to train on the new character classes. We purposely used small training sets for the new net to force it to rely as much as possible upon transferred knowledge as opposed to a large and varied training set to learn the new set of handwritten characters. Our results show a clear advantage in relying upon transferred knowledge to learn new tasks when given small training sets, if the new tasks are sufficiently similar to the previously mastered one. However, this advantage decreases as training sets increase in size.

Keywords: Knowledge Transfer; Deep Neural Nets, Inductive Transfer, Neural Nets

1. Introduction

All inductive learners require an appropriate bias. This allows sufficient generalization of a target concept with a reasonably sized set of training examples. An insufficiently biased learner will have an overly large hypothesis space to search. It will be prone to over-fitting and will require relatively large training sets. An overly biased learner will be able to learn only a poor approximation of the true target concept, no matter how many training examples it sees. A good bias will enable a learner to successfully acquire a concept with fewer training examples and greater generality.

In machine learning, bias is generally introduced in two ways. The first is through choice of the learning algorithm employed. This is succinctly stated in an introductory textbook as follows: “we will find it useful to characterize different learning approaches by the inductive bias they employ.” 1. The other common method is by controlling the examples used for training. Abu-Mostafa 2 pointed out that by adroitly choosing training samples, one
could ensure prior knowledge of the task to be learned, particularly in the form of desired
equivariances, would be correctly incorporated by a given machine learning algorithm.

Both of these approaches are sufficient for learners that live in a simple, static world and are only required to learn one task. This, however, is not the world we live in. The world through which we move requires us to learn continuously. Additionally, it does not tend to provide us with thousands of training examples for each and every task we must master. Nevertheless, we manage to learn.

We do this by fitting each new task to be learned into the framework of other relevant tasks, which we have already mastered. Our bias is that similar tasks will employ similar solutions and we will be learning many similar tasks. To the extent that two tasks (e.g. A & B) are related to each other, mastering task A makes learning task B easier. In some cases, mastering task A is actually necessary in order to learn task B (e.g. One must learn to walk before learning to jitterbug.). In other cases, the order in which tasks are learned is immaterial (e.g. Knowledge of Spanish helps when learning Italian; knowledge of Italian helps when learning Spanish. Neither language is a prerequisite for learning the other.).

Application of skills acquired in the process of mastering one task in order to aid in the learning of another is variously referred to as ‘knowledge transfer’, ‘transfer learning’ or ‘inductive transfer’. Here, we will refer to it as ‘knowledge transfer’.

In this paper, we examine knowledge transfer with deep convolutional neural nets (DCNN’s). We suggest the architecture of DCNN’s makes them particularly well-suited for knowledge transfer. Additionally, we show that when a generalized internal representation has been achieved, new concepts can be learned using both smaller training sets and nets with fewer free parameters. However, as the size of training set increases, net capacity becomes more important relative to transferred internal representation.

2. Background and Related Work

The work presented here draws from two streams of research - knowledge transfer and deep convolutional neural nets. We will begin by reviewing work done in knowledge transfer. Unless otherwise stated, all neural nets referred to are classic feed-forward 3-layer nets with the usual input layer/hidden layer/output layer architecture.

2.1. Knowledge Transfer

Some of the earliest successful work done in knowledge transfer dates back to the early 1990’s with Pratt’s work in discriminability-based transfer with neural nets. Previous work had attempted to use the weights learned by a net in solving one problem as the initial conditions for solving a new problem, but without success. The problem with this technique was that only some of the learned weights were useful for solving the new problem. Additionally, the weights that were unnecessary tended to interfere with the backpropagation learning algorithm because they tended to have large magnitudes. Pratt realized in order to overcome this obstacle, she had to first discover which nodes were not actually relevant for the new problem and reset their weights to relatively small magnitudes. This resulted in marked improvements in training time without any loss of accuracy.
This was an early demonstration not only of knowledge transfer, but also the importance of task similarity, since the more similar a new task was to previously mastered tasks, the more nodes would maintain their relevance, and therefore, aid in learning the new task. In fact, the whole basis of knowledge transfer is the assumption of some degree of similarity between different tasks, since otherwise there would be no point in transferring knowledge gained from one task to aid in learning another. One early method of grouping tasks by similarity is the Task-Clustering method, introduced by Thrun and O’Sullivan.

A technique very different than Pratt’s, called Multi-Task Learning, was developed by Caruana. The basic approach taken was to improve both generalization and learning speed by learning many related tasks simultaneously. This approach relied upon the various tasks being sufficiently related so they would constructively reinforce learning each other. In an earlier paper, Caruana lists several mechanisms for this reinforcement.

Although these two techniques both rely upon the creation of an internal representation of raw data to enhance a net’s ability to learn multiple tasks, they represent two fundamentally different methods of knowledge transfer - representational and functional. In representational transfer, which Pratt used, one finds the subsets of an existing internal representation which are useful for the new task being learned. In functional transfer, which Caruana used, one trains in such a way as to require learning a single internal representation, which is suitable for several tasks. Our technique, as will be shown, combines both approaches.

It was stated by Baxter that one way to view the problem of knowledge transfer is as the problem of learning an appropriate inductive bias for a set of tasks. If one can learn that bias while learning a subset of those tasks, then that bias can be used to aid in learning additional tasks from that set. This bias can take the form of choosing an internal representation, which provides a lower dimensional representation of the raw data for a specific task or group of tasks.

Additionally, Baxter points out that when attempting to learn a ‘family of concept learning tasks’, which is a potentially infinite set of mutually distinct binary classification problems (e.g. distinguishing among faces, distinguishing among characters, distinguishing among cars etc.) the real task is to learn a set of features that with an appropriate distance metric will enable one to tell if two inputs are close enough to be considered to be the same.

He used this approach to efficiently train a neural net to recognize Japanese kanji taken from a dataset from the CEDAR group at the State University of New York at Buffalo. His net consisted of 4 layers. The first layer was the input layer. The second layer contained the nodes that were trained to provide the common internal representation for all classes. The last two layers were grouped by task. Nodes in the third layer acted like a classical hidden layer, taking raw input and providing enough capacity for the given subnet to learn to recognize any individual kanji character. Each node in the final layer would each act as boolean output for a particular character, as determined by the subset of nodes in the third layer that provided it with input. He trained this net first to recognize 400 kanji. Then, he used the common internal representation to do 1-NN classification on 2,618 different kanji. He achieved a misclassification error of 7.5%.
A more recent example of learning an appropriate metric to distinguish among the elements of a set of classes was made by Chopra et al.\textsuperscript{11}. In this work, a siamese net\textsuperscript{12} was trained on a relatively small number of faces to recognize whether a given pair of faces were from the same person. This technique was then able to correctly label pairs of faces, which came from people not seen during training, as being same or different.

Like Chopra et al. and we use deep convolutional neural nets. Our net, which is shown in Figure 1, employs a slight variation of the LeNet 5 architecture first employed by LeCun et al.\textsuperscript{13}.

![Figure 1. Architecture of our net, which is a slightly modified version of LeNet5. It should be noted that the feature maps in layers C5 & F6 are 1 neuron × 1 neuron, which means they could with equal accuracy be considered as traditional neurons in a non-weight sharing feed-forward neural net.]

2.2. Deep Convolutional Neural Nets

This brings us to the second stream of research from which we draw - the use of deep neural nets, in particular, convolutional ones. Although these nets are usually referred to as just Convolutional Neural Nets, we refer to them as Deep Convolutional Neural Nets, in order to equally emphasize the use of a deep architecture as well as the properties that make them convolutional. Both of these properties are important for us.

Deep Convolutional Neural Nets (DCNN’s) are designed to limit the hypothesis space available to be searched. Specifically, they limit themselves to solutions that display\textsuperscript{13}:

1. Shift Invariance
2. Moderate Insensitivity to Rotations
3. Moderate Insensitivity to Geometric Distortions
Almost all image recognition problems require these properties. Because the net is architecturally restricted to hypotheses with these properties, there is no need for exhaustive training with hints, or extensive pre-processing of images.

These restrictions take the form of severely limiting the connections each node may possess and the weights associated with those connections. The nodes are organized into several feature maps. Their input only comes from one or more small groups of locally adjacent nodes in the previous layer. These groups are referred to as ‘receptive fields’. Each node will have no more than a single receptive field in an individual feature map. Furthermore, not only will neighboring nodes have adjacent or overlapping receptive fields, but they will also have the same input weights over those receptive fields.

This means in a given feature map, each node will react in exactly the same way to the same input from the layer below. Therefore, each feature map will detect and locate a given feature or combination of features from the previous layer. At higher layers of the net, feature maps may be regarded as identifying the combinations of various low-level features that comprise more complex higher-level features.

The presence of feature maps as an architectural component of DCNN’s makes these nets an attractive candidate for knowledge transfer, since these maps represent discrete, localizable detectors for specific features which distinguish among the various classes.

There are two standard types of layers found in a DCNN: convolutional and sub-sampling layers. In the convolutional layers (C-layers) of the net, adjacent nodes will have overlapping receptive fields with the same relative spatial positions. Because the weights associated with these fields are identical, calculating the output of a feature map in a C-layer is analogous to calculating the convolution of one or more small kernels over the feature maps of the previous layer. Although a single feature map may be connected to many feature maps of the prior level, it is connected to each by an individual learned kernel. This results in a feature map that will reflect the presence of a particular local feature, or local combination of features, wherever they occur in maps in the prior layer. It is this convolution of small kernels that gives DCNNs an architectural basis for translational invariance. It also provides a useful bias for problems with strong local correlations.

In the sub-sampling layers (S-layers), each feature map is connected to exactly one feature map of the prior layer. A kernel of the sub-sampling layers is not convolved over the corresponding feature map of the prior layer. Instead, the input feature map is divided into contiguous non-overlapping tiles, which are the size of the kernel. Each sub-sampling kernel contains two learnable parameters:

1. a multiplicative parameter, which multiplies the sum of the units in a given tile, and
2. an additive parameter, which is used as a bias.

This gives DCNNs a decreased sensitivity to minor rotations and distortions of an image, which helps make them robust with respect to unimportant variations.

In effect, the architecture of DCNN’s forces the early layers to act as feature extractors. Additionally, each layer of the net is learning an internal representation consisting of more complex features than the layer beneath it. So, the learning of simpler features is concentrated at the lower levels which act as pre-processed input to higher levels, which learn
more complex features. This should give successive internal representations with greater levels of specificity for a given task or set of tasks. For the sake of knowledge transfer, it also means that there will be multiple inductive biases from which to choose, allowing for various degrees of knowledge transfer between different sets of tasks.

The other advantage of DCNN’s is provided by their deep architecture. The most popular type of feed-forward neural net has three layers - an input layer, a hidden layer and an output layer. This architecture has achieved popularity because of its seeming simplicity and its ability to provide a large hypothesis space. In fact, it has been shown that arbitrary decision regions can be arbitrarily well approximated by continuous feed-forward neural networks with only a single internal hidden layer.

This would suggest that a three layer architecture is sufficient for any problem. However, it is not a terribly efficient architecture. One tends to need far more nodes to represent a given function with a shallow architecture than with a deep architecture. This would seem to imply that deep nets would be superior to shallow ones, because they need fewer parameters. However, training deep nets accurately seems to require specialized architectures or training techniques to avoid falling into local minima.

Fortunately though, there are techniques that avoid this pitfall. Some of these techniques, such as DCNN’s and Neural Abstraction Pyramid nets, rely upon their architectural restrictions to avoid the spurious minima which plague other deep nets. Others rely upon specialized training techniques, such as Cascade Correlation, Knowledge Based Cascade Correlation and Greedy Layer-wise Training.

It is our contention that DCNN’s, like all deep nets will have representations that move from lower level representations to higher level ones. When such a net is forced to learn a robust internal representation for a family of concept learning tasks, through functional transfer, then the learning of new tasks from the same family, may be readily accomplished through representational transfer. The hierarchical structure of the net will make the choice of which set of weights to transfer relatively straightforward without any in depth calculations. Furthermore, by using representational transfer to learn the new tasks, one will require far fewer samples to achieve reasonable accuracies.

3. Experiments

Our experiments involved training our net with the NIST Special Database 19, which contains 62 classes of handwritten characters corresponding to '0'-'9', 'A'-'Z' and 'a'-'z'. Our method was to first train the net on one subset of characters from the NIST database, using as many samples of each character as we could. Then, we would take a subset of different characters and see to what extent we could use the learned internal representations at various layers to enable the net to learn the new subset of characters with much smaller training sets and reasonable accuracy.

The decision to use very small sample sizes immediately raises the danger of overfitting. In fact, since we were only interested in using the first subset to learn an internal representation, it was very tempting to use all samples for training and none for testing. In trying to gain an appreciation for the extent to which overfitting would be a hazard,
we ran some experiments in which we tried to induce it. The results showed that, although marginally present, over fitting would not likely be a significant problem for us. This can be seen in Figures 2 & 3 which record some of our attempts to induce over fitting with both large and small training sets. We did this to simulate both the large training sets that would provide the knowledge we would attempt to transfer and the small training sets that we hoped would be sufficient to learn new concepts once we had knowledge to transfer.

Our first attempt to find the bounds of over fitting involved training with the first subset of characters. Here, we took a large sample set (7500 character images from a subset of 20 characters) and trained for 120 epochs to see what effects we would observe. Although, in early epochs, we saw a slight decline in accuracy, that seemed to correspond to over fitting, this decline soon rectified itself and after a brief high plateau of accuracy, the net seemed to settle into a fairly tight region of testing accuracy as training progressed. The random jitter within this region seems to be a larger effect than any systematic decline in accuracy. These results are shown in Figure 2. These results allow us to train on the entire first set of characters, without worrying about paying too high a price for over training.

We also tried to examine overtraining using a small set of characters. In Figures 2 & 3, we tried to over train using a training set with 20 images (1 sample of each character) and a testing set of 1000 images. Here, we could see with even more assurance that over training does not appear to cause significant degradation of net accuracy.

Because this result is counter-intuitive, it requires, at least, a good heuristic explanation.

The deleterious effects of over training are caused by two main factors:
Fig. 3. Training on a total of 20 samples, one from each of 20 character classes. The curve shown is the testing accuracy obtained on a set of 1000 samples drawn from that same character set. Even though perfect accuracy is achieved within the first 8 epochs, training is allowed to continue minimizing the net’s loss function. There is still no observable deleterious effects from over training.

1 using a model that is poorly biased - e.g. using a 10th degree polynomial to fit a quadratic function
2 using noisy training data

The data we used is surprisingly noise free. Although all the images are of hand written characters, none of the pen strokes are extraneous. They all are part of an individual person’s attempt to form a particular character. The main source of noise for us would be the rare instance of a mislabeled character.

Additionally, as we have previously stated, convolutional nets have several architectural restrictions, which give them a stronger bias than traditional feed-forward neural nets. The restriction each node has to restrict itself to a local receptive field severely limits the noisy coincidences that can be learned. Also, the restriction to shared weights places a higher cost on any learned feature, since it will have to either displace, or at least modify some other feature that could have been learned. These two characteristics make it less likely for a DCNN to learn a feature that is not broadly applicable.

The experiments we carried out were performed in the following way: The entire net was trained on a set of 20 characters using 400 samples of each. Each character was assigned a 20 bit random vector. The random bit assignment provides a greater distance between target vectors than a standard 1 of N encoding. The output of our net would be classified as belonging to the same class as the target vector to which it was closest according to a Euclidean distance measure. The resultant net was trained and achieved an accuracy of 94.38% on this training set. This net will be referred to as the ‘source net’,
since it will be the source of our transferred knowledge. Frequently, much larger training sets are used to obtain near perfect accuracy. However, for our purposes, this accuracy was deemed sufficient.

Next, we attempted to use some of the acquired knowledge to aid in learning to recognize a new set of 20 characters. These new characters were also assigned 20 bit random target vectors. Then, the weights from the bottom $n$ layers of the source net were copied over to the new net, where $0 \leq n \leq 5$. Transferred weights were kept fixed and not allowed to change during training. To find the best choice for $n$, we ran a series of experiments beginning with $n = 5$ and culminating with $n = 0$. This last scenario, of course, corresponds to the absence of any knowledge transfer. Were we to have tried allowing $n = 6$, that would correspond to transferring all the weights from the source net and not allowing any training. For obvious reasons, we did not do this.

The performance of each net was evaluated using a testing set comprised of 1,000 characters. We ran 5 learning trials for each value of $n$. Each trial was allowed to proceed until convergence was achieved on the training set. The accuracy we achieved on the testing set was then reported as the accuracy of our net. Additionally, we experimented with training sets of 1, 5, 10, 20 and 40 samples/class. Results are shown in Figures 4-7.

As each layer is released to be retrained, more free parameters become available, thus increasing the capacity of the net. However, the increase in free parameters is very sharply spiked at the 5th layer of the net. In fact, this is where more than 90% of the net’s free parameters lie. The number of free parameters for each layer is shown in table 1.

So, when only the top level has not been retained, the net can only train with 4.6% of
Fig. 5. Comparison of learning curves showing accuracy vs. number of retained levels for 1 sample per class in the training set. Curves show minimum accuracy, average accuracy and maximum accuracy obtained over 5 trials on a testing set with 1,000 character samples.

Fig. 6. Comparison of learning curves showing accuracy vs. number of retained levels for 10 samples per class in the training set. Curves show minimum accuracy, average accuracy and maximum accuracy obtained over 5 trials on a testing set with 1,000 character samples.

the free parameters normally available to it. When the top 2 levels are being retrained the net has 96.7% of its free parameters available for training.
Knowledge Transfer in Deep Convolutional Neural Nets

Fig. 7. Comparison of learning curves showing accuracy vs. number of retained levels for 40 samples per class in the training set. Curves show minimum accuracy, average accuracy and maximum accuracy obtained over 5 trials on a testing set with 1,000 character samples.

Table 1. Number of free parameters at each layer of our net.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Free Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>156</td>
</tr>
<tr>
<td>S2</td>
<td>12</td>
</tr>
<tr>
<td>C3</td>
<td>1,516</td>
</tr>
<tr>
<td>S4</td>
<td>32</td>
</tr>
<tr>
<td>C5</td>
<td>48,120</td>
</tr>
<tr>
<td>F6</td>
<td>2,420</td>
</tr>
<tr>
<td>Total</td>
<td>52,256</td>
</tr>
</tbody>
</table>

4. Discussion and Analysis

The main process being shown in Figure 4 is a trade-off between increased net capacity, as reflected by fewer retained levels/free parameters, and increased knowledge transfer as reflected by more retained levels. As more information is available to the net through increased training set size, the importance of transferred knowledge decreases and having sufficient capacity to learn the details of the new classes increases. It is also when only training the upper most levels that we see the only appreciable over fitting. This, we feel, is mostly due to the lack of the convolutional restrictions at the upper most layer, which causes the net to show no special favoritism for learning local features.

By observing the change in shape of the learning curves, it is possible to observe the increasing in importance of net capacity as the size of the training set grows. The shapes of
the learning curves for 1 sample/class, 10 samples/class and 40 samples/class may be seen in greater detail in Figures 5-7. The minimum, average and maximum accuracy obtained over each 5 trial run is shown to illustrate the relatively slight variance that was observed. These figures highlight the way in which each layer contributes to the transfer of knowledge from the source net. Furthermore, they emphasize the changing shape of the learning curve as the increase in training set size makes net capacity more important relative to transferred knowledge. This, however, may be taken with equal justification to be an indicator of the quality of the knowledge transferred. It seems likely that if the source net had learned either a larger set of classes, then the benefit of knowledge transfer would be greater and persist for even larger training sets. Perhaps a different set of classes, which in some sense spanned the set of classes better, would also give improved results.

It is interesting to observe how much of an advantage is obtained merely by retaining just the bottom four levels. Although these levels contain only 3.3% of the weights used by the net, their transfer leads to marked improvements in the accuracy of the net.

One may observe in Figure 4, that when levels C1-C5 are retained, the net doesn’t seem to have sufficient capacity to learn appreciably more information than is contained in about 10 samples. This implies that when attempting knowledge transfer between two DCNN’s, slightly more flexibility in choosing which weights should be retrained could be beneficial. For instance, perhaps, one could retrain only some of the feature maps at a given level, rather than all or none. This would enable us to have a partially transferred layer between the fully transferred and fully trained layers, which could help fine tune the balance between transferred knowledge and net capacity.

Lastly, one might also consider letting retraining take place with the transferred feature maps. This would undoubtedly give better results than were obtained, however, part of what we wanted to see was how much could be learned, without forgetting previously acquired concepts. One can now envision a particular sub-net being shared among several nets, each of which has been trained for different tasks.

5. Conclusions and Future Work

Our results show that for small training sets there is a clear advantage to favoring knowledge transfer over net capacity in both accuracy of learning and in effort required to learn.

What remains to be investigated is how quickly this trade-off changes as the number of classes that contribute to knowledge transfer increases. Baxter & Bartlett’s results strongly suggest that ultimately knowledge transfer will achieve accuracy comparable to the best achievable for full capacity nets with large training sets.

We plan to investigate methods of optimizing this trade-off, by allowing some feature maps at a given level to retrain. It should be possible adapt saliency, as in Optimal Brain Damage, to adapt mutual information, as in DBT, or to adapt Knowledge Based Cascade Correlation in order to determine which feature maps should be transferred and which should be retrained.

Finally, we will investigate techniques to select an optimal set of classes for knowledge transfer.
References