\documentclass [11pt,letterpaper ,twoside ,openany ]{report}

\title{Localizing Little Landmark with Transfer Learning}

\author{Sharad Kumar}

\usepackage{setspace}

\usepackage{geometry}

\geometry{margin=1in}

\usepackage{sectsty}

\chapternumberfont{\Large}

\chaptertitlefont{\huge}

\usepackage{graphicx}

\usepackage{subcaption}

\usepackage{float}

%\floatstyle{boxed}

%\restylefloat{figure}

\usepackage{chngcntr}

\counterwithout{figure}{chapter}

\counterwithout{table}{chapter}

\usepackage{wrapfig}

\usepackage{amsmath}

\usepackage{amsfonts}

%\usepackage{tabularx}

\usepackage{booktabs}

\usepackage{wrapfig}

\begin{document}

\maketitle

\tableofcontents

\begin{abstract}

In this work we explore the possibility of using context information to localize small objects in an image. To localize on the small object in a dog walking image - like location of hand-holding-leash, we create a regression model using Convolutional Neural Network (CNN) that is supervised by the coordinates of the small object in the image. Since small objects do not have strong visual characteristics in an image, it's difficult for neurons to discern their pattern because the feature map exhibit low resolution for small objects, which means much weaker signal for the neurons to recognize. Use of context for object detection and localization has been studied for a long time. This idea is explored by Singh \textit{et al.} in [1] for small object localization by using a multistep regression process where spatial context is used effectively to localize on the car door handle in the image of a car. We extend the idea in this paper and demonstrate that the technique of localizing in steps using contextual information when used with transfer learning can significantly reduce the training time.

\end{abstract}

\listoffigures

\chapter{Introduction}

\doublespacing

Because of complexity and variability in static images, semantic understanding of an image has remained one of the most difficult problem to solve in computer vision. While human eye is trained to rapidly cull relevant information from an image to build a language level understanding of the scene, such feat has remained elusive to artificial intelligence systems. Understanding relationships between objects in an image can help understand the semantic meaning of an image as it provides evidence for a particular situation being present in the image. For example, the ability of an artificial intelligence system to localize on a "hand-holding-leash" relationship in an image provides a fairly good clue that the image most likely has a dog and a person walking the dog, and this information can be used to generate semantic understanding of the image.

In this work we focus to localize on hand-holding-leash relationship in dog walking images. We only consider prototypical images for localization, where the image has only one dog and one dog-walker in the picture. Singh \textit{et al.} in [1] proposed the idea of using successively more relevant contextual information in a sequence of steps using a recurrent CNN architecture to localize on the small object. In this scheme, only the final step of the sequence is supervised by the location of the small object and predicts the target of small object location, while prior steps predict where to look next. The learning therefore discovers globally distinctive pattern to start the sequence and conditionally distinctive ones to get closer to the target in discrete steps.

Using Tensorflow\textsuperscript{\textregistered} we implement a model of the CNN model described in [1]. We use the car door handle dataset, provided by authors to train and test the model. We also train this model on our dog walking image dataset. We compare the results and report our findings in section 5.2. In section 4.3 we provide more details about the datasets we used for our experiments.

Our contribution in this work is to incorporate transfer learning to reduce training time for small object localization. The motivation of using transfer learning is the fact that generalizations learned by a model that has been trained on a large dataset can be effectively used as input activations to some other model. We propose a slightly different model by repurposing a pre-trained VGG-16 [2] model where we use activations from pool4 layer of the VGG-16 model. VGG-16 is a deep CNN model proposed by K. Simonyan and A. Zisserman from the University of Oxford that achieved 92.7\% accuracy in top-5 test category and 70.5\% in top-1 test category on Imagenet dataset. The Imagenet dataset [3] is a dataset of over 14 million images belonging to 1000 categories. We demonstrate significant improvement in training time with this approach and report our results in section 5.2.

\chapter{Related work}

\doublespacing

Localization of small object in an image using CNN is little understudied. Chen \textit{et al.} in [6] extend the R-CNN algorithm to detect small objects - like computer mouse on a desk, or a faucet in a kitchen. In their approach, they use a modified Region Proposal Network (RPN) [7] by choosing object proposals many times smaller than used in the original RPN. To use context information, they also crop a region centered at region proposal, but bigger than the region proposal. The region proposal and the context proposal are fed to two parallel CNNs and their concatenated activation are used as input to a third CNN to make predictions. In this work authors use Intersection over Union (IoU) as performance metric whereas in our work we use Euclidian distance between original and predicted coordinated normalized by the bounding box of the object as explained in section 4.3.

Another important work for small object identification is done by Eggert \textit{et al.} in [8] where they modify Faster R-CNN model [7] that leverages higher-resolution feature maps for brand logo detection. Their work qualifies as small object detection since they are trying to locate brand logos in pictures that were not intended to capture it - like image of a soft drink brand in a picture taken at an outdoor concert venue, or image of a sport brand on a person's shirt who is walking a dog - and therefore tend to occupy small image area. In their work they attempt to generate better region proposals and assume a perfect classifier. They compare performance of region proposal generator using activations from conv3, conv4 and conv5 layers of pre-trained VGG-16 model [2]. They found that conv3 and conv4 layers' activation performed better than conv5 layer activations.

The paper that we extend from on this work [1] proposes an architecture that is recurrent in the sense that the feature map generated by one step of the model is encoded as contextual information and fed as input to the next step in the sequence along with feature map generated by the convolutional layer. Another important work that explores this idea is by Zuo \textit {et al.} in [9]. In their work they argue that convolutional and pooling layers in a CNN are performed locally without considering other regions of the image and therefore fail to capture contextual dependencies for better representation. They propose a model that encodes this correlation for better performance.

We use transfer learning to demonstrate that instead of training a network from scratch, using a pre-trained network may reduce training time significantly. Pan \textit {et al.} in [10] do an in depth study of feasibility of transfer learning and show that knowledge learned by a model in one domain can be transferred to another machine learning model in a different domain even when the feature space and/or the data distribution of source and target systems is not the same. Shin \textit {et al.} also reiterate this idea in [11] where they employ transfer learning to fine-tune a CNN model pre-trained on natural image dataset (RGB) to medical image (monochrome) classification task. This idea has been successfully used by numerous researchers and practitioners in image classification and localization tasks by using patterns learned by deep CNN models trained on enormous amount of images.

\chapter{Deep Learning/Convolutional Neural Network(CNN)}

\doublespacing

Since its resurgence in 2012 when a neural network based architecture, called AlexNet was proposed by Krizhevsky \textit {et al.} [26] for image classification of ImageNet [3] dataset, neural networks variant deep Convolutional Neural Networks (CNN) have achieved significant improvement in state-of-the-art for classification as well as regression tasks. A CNN model consists of many layers (Figure 1) and each layer learns features or representations at increasing level of abstraction, as demonstrated in [4, 5] using a deconvnet that map these representations back to the input pixel space. CNN were created to overcome the scaling problem of traditional neural networks. In traditional neural networks, neurons in one layer of the network are connected to all the neurons in adjacent layers, which makes it difficult to scale for intelligent image understanding and analysis. Neurons (also called filters or kernels) in each layer of the CNN are connected only to a small region of the input and are three dimensional (Figure 2).

\begin{figure}[h]

\centering

\begin{subfigure}[b]{0.90\linewidth}

\includegraphics[width=\linewidth]{Images/Figure1-CNN}

\end{subfigure}

\caption{Convolutional Neural Net}

\label{fig:cnn}

\end{figure}

\begin{wrapfigure}{l}{0.5\textwidth}

\begin{center}

\includegraphics[width=0.48\textwidth]{Images/Figure2-CNNFilter}

\end{center}

\caption{CNN Filter}

\end{wrapfigure}

The height and width of the filter determines the region of spatial connectivity of the filter to a layer before. Depth of the filter indicates how many such filters are there in what's called a filter bank. A filter contains real numbered weights. The dot product of these weights with the input region that it's spatially connected to creates output activation map (also called feature map), as depicted in Figure 2. A filter convolves (slides) across it's input with a predefined stride. As illustrated in Figure 1, the input is zero padded if the filter does not align with the image at the edges as it convolves. The weight in the filter remains unchanged or is shared as convolutions are performed in a layer. The rationale of sharing weights is twofold: first it reduces the number of parameters in the model, and secondly any useful features identified in one part of the image can be re-used everywhere else without having to be independently learned as explained by Murphy in [27].\\

\noindent

\textbf{\textit{Forward Pass}} \\

There are mainly three types of components in a CNN layer - convolution, Rectified Linear Unit (ReLU) and pooling - and the layers are arranged in a sequence, as depicted in Figure 1. Each CNN layer transforms one volume of activations into another volume during the forward pass of processing. Filters are mainly involved in computation in the convolution layer and process its input as explained above. ReLU layer introduces non-linearity into the model by applying \(max (0,x)\) function element-wise to the activations produced by the filters in the convolution layer. Use of \(max (0,x)\) non-linear function results in accelerated training as compared to logistic or hyperbolic tangent function as explained by Krizhevsky \textit {et al.} in [26], where the gradient in the saturating part of the activation function graph becomes very small (for very large or very small weights) resulting in sluggish training because of minuscule weight change during back-propagation. Introducing non-linearity with ReLU ensures that the CNN model does not collapse into a large linear model [27, 12]. Neural networks can therefore be thought of as a nonlinear generalization of the linear model, which by introducing the nonlinear transformation greatly enlarges the class of linear model [12].

\begin{wrapfigure}{r}{0.6\textwidth}

\begin{center}

\includegraphics[width=0.48\textwidth]{Images/Figure3-Pooling}

\end{center}

\caption{Pooling}

\end{wrapfigure}

Activations from the ReLU layer are subsampled by the pooling layer. As shown in Figure 3, pooling layer subsamples the activations spatially and is usually done either by averaging or by computing the max over a small window of activations produced by the ReLU layer. Subsampling reduces the number of parameters in the model and hence the amount of computation in the network, it also results in small shift invariance as the image activations propagate through the network. Because of the subsampling, there's a progressive reduction in size of activations, as shown in Figure 1. \\

\noindent

\textbf{\textit{Back Propagation}}\\

Weights in the filters are the parameters of a CNN model and model fitting involves learning these weights using back-propagation. Back-propagation is a optimization algorithm where the goal is to find a set of value for model parameters that generalizes the learning in such a way that the model achieves superior classification or regression accuracy on data that it has not seen before. This goal is achieved by minimizing the error on training dataset by using gradient descent (or its variant) method for optimization. Validation dataset complements the training dataset to find training hyper-parameters that generalizes learning. Stochastic gradient descent is one of the most effective techniques for back-propagation where gradient of the cost function with respect to weights and the chain rule of derivatives are used to update model weights iteratively in minibatches. Use of minibatches in stochastic gradient descent accelerates the training and also helps to obtain unbiased estimate of the gradient by taking the average gradient on a minibatch, assuming examples in minibatches are drawn in identically and independently distributed (IID) fashion as explained by Goodfellow \textit {et al.} in [13].\\

\noindent

\textbf{\textit{Overfitting and regularization}}\\

Because of large number of parameters, CNN models have proclivity to overfit when not designed with suitable regularization parameters or when there is not sufficient data to train the model. Large number of parameters in a neural network may result in a model that shows high variance when the model is trained on different training datasets as explained by Hastie \textit {et al.} in [12]. Overfitting is a manifestation of this problem in neural network where training fails to achieve good model generalizations resulting in good training accuracy, but poor performance on test dataset. Bias in a model is the other source of prediction error which is usually caused by wrong selection of model for the task. In case of overfitting, variance rather than bias dominates the estimation error. Regularization addresses the problem of overfitting by reducing variance significantly while not increasing the bias as explained by Goodfellow \textit {et al.} in [13]. \(L\_1\) and \(L\_2\) are two most commonly used regularization techniques in machine learning in general, including neural networks. Dropout a technique proposed by Hinton \textit {et al.} in [14] is another regularization technique used specifically with neural network. In our implementation of localizing on dog walking image, we explored \(L\_1\), \(L\_2\) regularizations as well as dropout and we report our findings in the experiments section 5.3. In [1] authors use \(L\_2\) regularization as we explain in section 4.3.

\chapter{Detecting Small Object in an Image}

\doublespacing

An efficient object detection model in an image may use region proposals, as one used by Girshick \textit {et al.} in R-CNN [7, 15]. Their two stage architecture is current state-of-the-art where the first stage extracts region proposals and the second stage uses these region proposals as input to a CNN which is pre-trained on an auxiliary dataset. This setup has shown remarkable performance improvement for prominent object detection in an image, but is not suitable for small landmarks as small objects do not have very distinctive features. A simple solution to address the issue of lack of distinctness for little landmark in an image can be either to magnify the image or take a high resolution image. As explained by Eggert \textit {et al.} in [8], this simple approach may not be very effective since computing convolutions in CNN grows quadratically with image dimensions and will result in unnecessary computation. Moreover, as explained by Chen \textit {et al.} in [6], low resolution inputs for small objects are deeply embedded in the nature of visual perception and a robust vision system should be able to deal with it.

This limitation of R-CNN beckons a different deep learning architecture to localize small objects in an image. Singh \textit {et al.} in [1] propose a stepwise regression model for this task and they call this model an architecture for localizing \textit {little landmark}. They propose a recurrent model that is trained end to end using location of the little landmark in an image. Authors demonstrate the robustness of this model by testing and training it on several datasets. One of contribution of their work was to create and annotate a dataset with images containing little landmarks. Authors use Stanford's car dataset [15] for this task and annotate the dataset with location of the car door handle for little landmark localization. We use this Car Door Handle (CDH) dataset and our own Dog Walking Images (DWI) dataset for training and testing our models. Details of these datasets are provided in section 4.3.

\section{Architecture Overview of Little Landmark CNN Model}

The CNN architecture for small object detection in [1] exploits the fact that a little landmark is defined by it's context. Authors define \textit {latent landmark} as a location that are more prominent than the little landmark and can be used as context to detect the little landmark conditionally. The localization in this architecture happens in a sequence of steps where a series of latent landmarks lead to the target little landmark by providing contextual information. Figure 4 illustrates this process with two examples. In the image on the left hand side, the car door handle is detected in a sequence of three steps where a trained network finds the first latent landmark (in red) near the front wheel, the second latent landmark in the bottom half of the front door (in green) which helps find the target location (in blue) in the third step. Similar pattern can be seen in the image on the right hand side in Figure 4 where edge of the electrical switchboard is used as the starting point to localize on the electrical switch on a wall in three steps.

\begin{figure}[h!]

\centering

\begin{subfigure}[b]{0.45\linewidth}

\includegraphics[height=4.5cm, width=6.75cm]{Images/Figure4a-CDH}

\caption{Car Door Handle Localization}

\end{subfigure}

\begin{subfigure}[b]{0.45\linewidth}

\includegraphics[height=4.5cm, width=6.75cm]{Images/Figure4b-LS}

\caption{Light Switch Localization}

\end{subfigure}

\caption{Localization in three steps}

\label{fig:coffee}

\end{figure}

Figure 5 shows a schematic diagram of the little landmark localization architecture that localizes on a car door handle in three steps. In this recurrent CNN architecture, the bottom convolutional layers are used to extract image features which is common input to all steps. Three columns in the upper portion of the figure corresponds to the stepwise procedure that detects little landmark in the last step. Every prediction step \textit {s} in the model generates location of the latent landmark (red blob) and predicts the location of next latent landmark (blue blob). The prediction for next step is encoded as a feature map \(P^{(s)}\) and along with image features is used as input to the next step. Loss calculation for the model is explained later in section 4.3, but it's worth noting here that the difference between prediction (blue blob) in step \textit {s-1} and the generated latent landmark location (red blob) in step \textit {s} contributes to total error of an iteration.

\begin{figure}[h]

\centering

\includegraphics[width=12cm, height=12cm]{Images/Figure5-Architecture}

\caption{Little Landmark Architecture}

\label{fig:arch}

\end{figure}

Weights in row two and three of \textit {s} steps are shared. It must be noted that this CNN model does not use the pooling layer. Although authors do not explain why, but we surmise that since pooling will result in a low resolution feature map than the original one, it may make identification of little landmark more difficult as it already has non-distinctive characteristics.

\section{Image Pre-processing}

A batch of ten images is created as input to the model. Image height and width are adjusted so that all images in a batch have same size. When necessary, images are zero padded to match the longest width and longest height among all images in a batch. Images are normalized by performing mean subtraction and standard deviation scaling. Random scale jitter is added to images and image are also randomly flipped to introduce noise to ensure that model does not overfit on training data.

A location grid, \textit{\texttt{out\\_locs}}, is created for the batch which corresponds to spatial dimension of the output from the CONV/RELU6 layer of the bottom portion of the model. This location grid is used for generating the latent landmarks (red blobs in figure 5) as explained later in next section.

\section{Prediction Model}

Location of target little landmark is the only supervision used in the model and this model learns to infer latent landmarks. As explained before, each step in the model learns to find its latent landmark and generates prediction about the location of next latent landmark for the next step. We explain this process next.\\

\noindent

\textbf{\textit{Latent Landmark Centroid Generation (red blob in figure 5)}}\\

The CONV/RELU11 layer of step \textit {s} produces an output of dimension \textit {w} x \textit {h} x \textit {26}, where value of \textit {w} and \textit {h} is determined by the width and height of images in input batch. The output therefore has 26 \textit {w} x \textit {h} dimensional layers. The product \textit {l} = \(w\*h\) corresponds to the location grid, \textit{\texttt{out\\_locs}}, created during pre-processing stage. Network output \(z\_l^{(s)}\) from \textit {l} locations of the first layer of \textit {w} x \textit {h} x \textit {26} dimensional output is used to calculate the latent landmark \(pc\) of step \textit {s} as follows

\[ pc^{(s)} = \displaystyle\sum\_{i=0}^{l} q^{(s)}\_i \* \textit{out\\_locs}\_i \tag{1}\]

\noindent

where \(q\_i^{(s)}\) is softmax over all locations computed as \(q\_i^{(s)} = e^{z\_i^{(s)}}/\ \Sigma\_i e^{z\_i^{(s)}} \) and \(z\_i^{(s)} \in \mathbb{R} \) is the output from network for confidence at \(l\_i \)\\\\

\noindent

\textbf{\textit{Latent Landmark Prediction for Next Step (blue blob in figure 5)}}\\

Each step produces an estimate of the next step's latent landmark, \(P^{(s)}\). \(pc^{(s)} \) calculated in (1) along with prediction \(poc\) made by each location \(l\_i\) is used for generating \(P^{(s)}\). \textbf{The sum \(pc + poc\) is prediction of centroid for next step's latent landmark, except for the last step. In the last step, \(pc + poc\) predicts the location of the little landmark}.

To calculate \(poc\), a logical grid of \textit {5} x \textit {5} cells is placed over each location \(l\_i\). Last 25 layers of \textit {w} x \textit {h} x \textit {26} dimensional output from CONV/RELU11 is used for this calculation. This scheme is illustrated in Figure 6.

\begin{figure}[t]

\centering

\includegraphics[width=\linewidth]{Images/Figure6-Grid}

\caption{Latent Landmark Prediction Grid}

\label{fig:grid}

\end{figure}

Left hand side of Figure 6 shows 25 logical grid points \(g\_j\) corresponding to a locations \(l\_i\) (as shown in right side of Figure 6), where \(g\_j(x), g\_j(y) \in \{-50, -25, 0, 25, 50\} \) pixels. The network produces 25 confidence values \(o\_{ji}^{(s)}\) for \(j \in \{1, ..., 25\} \) from last 25 layers of output from CONV/RELU11 layer. These \(o\_{ji}^{(s)}\) values are softmax of network outputs. Each location \(l\_i\) produces the estimate of next latent landmark as follows

\[ p\_i^{(s)} = \displaystyle\sum\_{j=0}^{25} o\_{ji}^{(s)} \* g\_j \tag{2} \]

\noindent

These \( p\_i^{(s)} \) values are used with confidence values \(q\_i^{(s)} \), produced by first layer of the output from CONV/RELU11 layer, to produce \( poc \) as follows

\[ poc^{(s)} = \displaystyle\sum\_{i=0}^{l} q\_i^{(s)} \* p\_i^{(s)} \]

\noindent

Finally, to generate prediction \(P^{(s)} \) for latent landmark of the next step, encoding is done by placing a radial basis kernel with \(\beta \) = 15, centered at \( pc + poc \). In the last step, \( pc + poc \) predicts the location of the little landmark - coordinates of the car door handle in figure 5.\\

\noindent

\textbf{\textit{Training (and Loss Calculation)}}\\

We train the model through back-propagation using Adam optimizer. \(L\_2\) loss between predicted location and the ground truth is used for gradient calculation. However, there are several components in loss value. \(L\_2\) loss between the ground truth and the location predicted by the last step of the model and the \(L\_2\) loss between latent landmark predicted in step \(s\) and estimated in step \textit {s-1} contribute to the total loss of the model in one iteration, which is averaged over the minibatch of ten images to perform back-propagation of error through the network. The weightage given to the step losses is less than the weightage given to \(L\_2\) loss between ground truth and predicted location in the last step as shown below

\[ L = \displaystyle\sum\_{s=1}^{S} \lambda \_s L^{(s)} + R(\theta)\]

\noindent

We use \( \lambda \_s = 0.1 \), except for the final step \(S\) where \( \lambda \_s = 1\). \( R(\theta) \) is a regularizer for the parameters of the network. We use \(L\_2\) regularization of network weights with a multiplier of \(0.00005\).

Incorporating loss from all steps of the model in the loss function encourages earlier steps to be informative for the later steps by penalizing disagreement between the predicted and later detected latent landmark locations. Also, encoding the prediction as a feature map instead of as a rigid constraint for the next step allows it to ignore the prediction from the previous step if necessary. This flexibility is helpful in early stages of the training when latent landmark estimates are not very reliable.\\

\noindent

\textbf{\textit{Performance Metric}}\\

To evaluate model accuracy, we use a generally accepted [17] metric of plotting detection rate against normalized distance from ground truth. The \(L\_2\) norm between actual and predicted location is normalized by the height of the bounding box of the car in car door handle dataset, and by height of the switchboard in light switch dataset.\\

\noindent

\textbf{\textit{Datasets}}

\noindent

\textit{Car Door Handle (CDH) Dataset}\\

We received an annotated dataset of CDH dataset from authors of [1]. There were 1920 training images and 1200 testing images. All images had the coordinates of the car door handle annotated. Car bounding box details were also available.\\

\noindent

\textit{Dog Walking Images (DWI) Dataset}\\

We used our own dog walking images to test model's effectiveness on a different dataset. We use 310 images for training and 70 for testing. We annotated the coordinates of hand-holding-leash part of the image. The leash bounding box was already annotated in our dataset.

\chapter{Implementation of Little Landmark Model}

\doublespacing

Authors of original paper provided us their Matlab implementation for the model. We used that codebase to create our own implementation of the model using TensorFlow\textsuperscript{\textregistered} and Python. We tested fidelity of our implementation by comparing our test results with the results authors published in their paper [1]. We used the same dataset as the authors' used for training and testing the model and found that our model performed similar to authors'. Figure 7 shows a comparison between the models. \\

\begin{figure}[h!]

\centering

\begin{subfigure}[b]{0.49\linewidth}

\includegraphics[height=4.5cm, width=6.75cm]{Images/Figure7a-Original}

\caption{Original}

\end{subfigure}

\begin{subfigure}[b]{0.49\linewidth}

\includegraphics[height=4.5cm, width=6.75cm]{Images/Figure7b-TF}

\caption{TensorFlow}

\end{subfigure}

\caption{Original vs. Tensorflow}

\label{fig:coffee}

\end{figure}

The plot of normalized distance vs. detection rate from our Tensorflow implementation is depicted in Figure 7(b). Pred 3 plot in Figure 7(a) shows authors' model accuracy. Pred 2 and Pred 1 plots in Figure 7(a) are of model with two steps and one step respectively.\\\\

\noindent

\textbf{\textit{Training Detail}}\\

We tested the Tensorflow\textsuperscript{\textregistered} model on a Linux computer with a GPU. We trained the model for 3100 epochs where each epoch trained on a shuffled set of 1920 images in batches of ten images. The training ran for about 107 hours.

\section{Model Modifications and Experiments}

\noindent

\textbf{\textit{Transfer Learning}}\\

Fergus \textit {et al.} in [4] and many other researchers [18, 19] have demonstrated that in a well-trained CNN model lower layers get attuned to more generic feature of an image like edges, corners or color blobs whereas higher layers learn more specific features like a person's face or wheels of a car. Bengio [18] argues that a deep learning algorithm seeks to discover good representations at multiple layers in such a way that features learned in higher level can be composed of features learned in lower layers of the network. Another important characteristic of learning in this scheme is that features learned in higher layers are invariant to variation in training distribution - like background, viewpoint, scene context etc. Fergus \textit {et al.} [4] by mapping activations in a CNN back to pixel space experimentally demonstrate this fact as shown in Figure 8. Properties of a deep CNN discussed above make transfer learning possible where representations learned with a CNN trained on a large image dataset can be effectively used to initialize another network. Oquab et al. [19] show that despite differences in image statistics and tasks in the two datasets, the transferred representation leads to significant improvement in image detection and classification as incorporation of prior knowledge via transfer learning can boost the performance of CNNs by a large margin. Because of these advantages two most common reasons to employ transfer learning is that it can reduce training time and that it can overcome overfitting problem when there is paucity of training data.\\

\begin{figure}[h]

\centering

\begin{subfigure}[b]{0.22\linewidth}

\centering

\includegraphics[height=3cm, width=3cm]{Images/Figure8b-Deconv}

\caption{Layer 2 activation mapped to pixel space}

\end{subfigure}

\begin{subfigure}[b]{0.22\linewidth}

\centering

\includegraphics[height=3cm, width=3cm]{Images/Figure8a-Deconv}

\caption{Image patch corresponding to (a)}

\end{subfigure}

\begin{subfigure}[b]{0.22\linewidth}

\centering

\includegraphics[height=3cm, width=3cm]{Images/Figure8d-Deconv}

\caption{Layer 2 activation mapped to pixel space}

\end{subfigure}

\begin{subfigure}[b]{0.22\linewidth}

\centering

\includegraphics[height=3cm, width=3cm]{Images/Figure8c-Deconv}

\caption{Image patch corresponding to (c)}

\end{subfigure}

\caption{DeconvNet}

\label{fig:devonv}

\end{figure}

\noindent

\textbf{\textit{Little Landmark with Transfer Learning}}\\

Our motivation to use transfer learning was to exploit generalizations learned by VGG-16 network [2] on ImageNet [3] dataset to reduce the training time of for little landmark localization. As discussed before, the model proposed in [1] takes more than 100 hours to train. We use transfer learning to improve training time for this model where we use activations from another state-of the-art CNN model, pre-trained on an auxiliary dataset, as the starting point of our model training. We observed significant improvement in training time in out implementation that used transfer learning. We evaluated out model accuracy on car door handle dataset and saw comparable results.

\begin{wrapfigure}{l}[9mm]{0.5\textwidth}

\begin{center}

\includegraphics[width=0.48\textwidth]{Images/Figure9-DWI}

\end{center}

\caption{Dog Walking}

\end{wrapfigure}

At Portland State University we have our own dataset of dog walking images that we use for out computer vision projects. It's a small dataset and its statistics are explained in section 4.3. Figure 9 shows a typical image from the dataset. Understanding situations in an image is an ongoing project at Portland State University. Visual situations are concepts such as ``a crowd waiting for a bus,'' or ``a game of ping-pong,'' or ``walking the dog,'' [20]. Detection of such situations in an image helps in semantic comprehension of the image. Another goal in our work was to localize on ``hand-holding-leash'' part of the image. The rational was that detection of ``hand-holding-leash'' situation in the image will give more evidence that a person may be walking a dog in the image. This detection will help localize on other objects or situations in the image and in general semantic understanding of the image. To this end, we used the design proposed in detection of little landmarks [1] to localize on hand-holding-leash part of the dog walking image.

We used transfer learning to overcome the problem of overfitting because of scarce training data - we did encounter overfitting problem when we trained the model from scratch with limited data in dog walking image dataset. \\

\noindent

\textbf{\textit{Little Landmark Model Modification using Transfer Learning}}\\

As explained before, we used generalizations learned by VGG-16 model for training the car door handle as well as the dog walking image datasets. VGG-16 [2] is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford that achieved 92.7\% accuracy in top-5 test category and 70.5\% accuracy in top-1 category on Imagenet dataset. Imagenet dataset [3] is a dataset of over 14 million images belonging to 1000 categories.

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Images/Figure10-VGG-16}

\caption{VGG-16}

\label{fig:vgg}

\end{figure}

Figure 10 shows a schematic diagram of VGG-16 model. VGG-16 CNN architecture has sixteen weight layers (model D on the right hand side in Figure 10) and uses a filter of size 3 x 3 to convolve in all layers. The input to the network is a fixed size 224 x 224 RGB image. Mean image subtraction is the only preprocessing performed on input images to center the images. We use activations from Pool4 layer of the model (layer with dimensions 14 x 14 x 512 on left hand side in Figure 10) as input to our modified little landmark model as shown in Figure 11.

\begin{figure}[h]

\centering

\includegraphics[width=12cm, height=9cm]{Images/Figure11-Modified-LL}

\caption{TL Architecture}

\label{fig:tl\_arch}

\end{figure}

In our modified architecture we use a bridge layer that takes Pool4 activations from VGG-16 model as input and performs a layer of convolution to modify the output dimensions so that it conforms with rest of the architecture. We used a publicly available pre-trained VGG-16 model from Github [21] as the basis of our model. We used same image preprocessing that we used for the original little landmark model and fed a batch of ten images to VGG-16 model, as before. Internally the VGG-16 implementation warped images to 224 x 224 x 3 before feeding it to the network.

\section{Experiments and Results}

\noindent

\textbf{Experiment 1}\\

In this experiment we used our modified model that uses VGG-16 model with transfer learning (Figure 11). We trained the model on same car door handle dataset as the original model and found that performance of both models were comparable. However, our modified model needed about 46 hours of training for 530 epochs. In this training we fixed the weights of the pre-trained VGG-16 model and fine tuned only the weights in regression steps of the model. We saw a 57\% improvement in training time of the model. Figure 12(a) shows the normalized distance vs. detection rate plot for this experiment.

\begin{wrapfigure}{l}[9mm]{0.5\textwidth}

\begin{subfigure}[b]{\linewidth}

\includegraphics[height=5cm, width=7.5cm]{Images/Figure12a-Exp1}

\caption{Experiment 1}

\end{subfigure}

\begin{subfigure}[b]{\linewidth}

\includegraphics[height=5cm, width=7.5cm]{Images/Figure12b-Exp1}

\caption{Experiment 2}

\end{subfigure}

\begin{subfigure}[b]{\linewidth}

\includegraphics[height=5cm, width=7.5cm]{Images/Figure12c-Exp1}

\caption{Experiment 3}

\end{subfigure}

\caption{Experiment Results}

\end{wrapfigure}

\noindent

\textbf{Experiment 2}\\

In this experiment we trained the same network as we used in experiment 1 with dog walking images. 310 training and 70 test images were used in this experiment to localize on hand-holding-leash part of the image. We trained the model for 96 hours and saw near perfect training accuracy. However, we saw very poor test accuracy as shown in Figure 12 (b).

The model appeared to overfit by memorizing the features of training images and did not seem to learn any general pattern for localization. We trained the model with weight loss values of 0.005 and 0.0005 also instead of 0.00005 that we used for \(L\_2\) regularization in the original model. We also experimented with \(L\_1\) regularization. None of these method resulted in any performance improvement. We also used dropouts [14] for regularization, but did not see any performance improvement.

Since context plays an important role in little landmark's localization we changed the model to localize in five steps instead of three. Our rational for this approach was that a model localized in five steps may pick up better context which may result in better generalization. However, this approach also did not work.

Another unsuccessful attempt to improve performance was to use activations from Pool3 layer of the VGG-16 model. The rational for this approach was that since latent and little landmarks are both not very distinctive, using filters from a lower layer, that may be more attuned to features of latent and little landmarks, may show better performance. \\

\begin{wraptable}{r}{0.55\textwidth}

\centering

\tiny

\begin{tabular}{ | p{1.4cm} | p{1.4cm} | p{1.4cm} | p{1.4cm} | p{1.4cm}| }

\hline

%Normalized Distance & & Detection Rate & & \\ \hline

Normalized Distance & \multicolumn{4}{|c|}{Detection Rate} \\ \hline

& Original & Experiment1 & Experiment2 & Experiment3 \\ \hline

0.01 & 0.29 & 0.28 & 0.00 & 0.01 \\ \hline

0.02 & 0.71 & 0.49 & 0.00 & 0.02 \\ \hline

0.03 & 0.81 & 0.73 & 0.00 & 0.12 \\ \hline

0.04 & 0.86 & 0.83 & 0.00 & 0.23 \\ \hline

0.05 & 0.90 & 0.91 & 0.00 & 0.44 \\ \hline

0.06 & 0.88 & 0.94 & 0.14 & 0.46 \\ \hline

0.07 & 0.90 & 0.94 & 0.14 & 0.65 \\ \hline

0.08 & 0.93 & 0.95 & 0.14 & 0.66 \\ \hline

0.09 & 0.93 & 0.97 & 0.43 & 0.75 \\ \hline

0.10 & 0.97 & 0.98 & 0.43 & 0.75 \\ \hline

\end{tabular}

\caption{Experiment Results}

\label{table:exp}

\end{wraptable}

\noindent

\textbf{Experiment 3}\\

In this experiment we trained the original TensorFlow model that we explained in section 4.3, but with same amount of training data that we used for experiment 2. We anticipated that training the original little landmark model with small amount of data may render the model not as effective. For about the same duration of training, we did see inferior performance in this experiment, but the performance was not as poor as in experiment 2. Figure 12 (c) shows normalized distance vs. detection rate plot for this experiment.

Table 1 shows a comparison of all model performances.\\

\chapter{Conclusion}

\doublespacing

As we discussed before little landmarks in an image do not have very distinctive features and use of context is an effective technique for their detection. We demonstrated this by implementing the architecture originally proposed by Singh et al. [1] and also by modifying their design to incorporate transfer learning. By using transfer learning we were able to show significant improvement in training time while not degrading model performance. We also tried to exploit transfer learning to overcome the problem of overfitting because of insufficient training data. Although many researchers [19, 22, 23] have been successful in this endeavor, we did not see that upturn in experiment 2. We next explain why this approach did not work in our case.\\

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Images/Figure13-CDH}

\caption{Representative Images from Car Door Handle Dataset}

\label{fig:cdh}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=\linewidth]{Images/Figure14-DWI}

\caption{Representative Images from Dog Walking Images Dataset}

\label{fig:dwi}

\end{figure}

The comparison of results between experiment 1 and experiment 3 (Table 1) demonstrates that lack of data does impact a model accuracy. However, comparison between experiment 2 and experiment 3 shows that lack of training data resulted in much worse performance in dog walking images as compared to cars images. Figure 13 shows some representative images of the cars used in training to localize on the car door handle. In all these images location of the door handle has such a regular spatial configuration that latent landmarks can be consistently found, in other words the context for little and latent landmarks are consistent across images. We discovered that for a three-step localization, the first latent landmark is found near the front or rear wheel, the second in the door, which helps in target detection. Whereas in dog walking images (Figure 14), the context around the hand-holding-leash landmark is not consistent between images and as a result the model does not train as well as the car door handle dataset model does.

\chapter{Future Work}

\doublespacing

A hallmark of CNN models is the ability to learn invariant features of an image so that the prediction or classification accuracy is not impacted by variability in an image. Xu \textit {et al.} explain [24] that representations in very top layer of convnet distills information in image down to the most salient objects. However this property of CNN is not very useful for tasks like small object detection since they are defined more by context. They propose an attention based model in [24] where they argue that using low level representations may help preserve the semantic information in an image. In their work they draw an parallel between this approach and presence of attention in human vision system. They use this insight to create a Long Short Term Memory (LSTM) based CNN system that automatically learns to describe the content of images. We argue that same approach of using attention based model can be explored for identification for small objects in an image.

The variant of transfer learning that we used in our work is supervised transfer learning where the source network was trained with supervision. Another form of transfer learning uses unsupervised pre-training. Erhan \textit {et al.} explain in [25] that greedy layer-wise unsupervised pre-training overcomes the challenges of deep learning by introducing a useful prior to the supervised fine-tuning training procedure, which effectively puts parameter values in the appropriate range for further supervised training. Use of this approach makes unsupervised learning a special kind of regularizer that minimizes variance at the cost of introducing some bias in the model. This is another promising method for small object localization specially in cases where training data is scarce and context around the small object is not consistent.

Another slightly unrelated future work is development of better metric to measure effectiveness of small object detection. Chen \textit {et al.} discuss in [6] that one of the challenges of small object identification is that we do not know how difficult this task is or how well do existing object detectors work. We found that metric used for measuring performance of small object detection is not consistent - some researchers use Intersection over Union (IoU) whereas others use some sort of normalization for Euclidian distance between prediction and ground truth. A good metric therefore can be very useful tool in this area or work.

\begin{thebibliography}{9}

\bibitem{Singh2016}

Saurabh Singh, Derek Hoiem and David Forsyth,

\textit{Learning to Localize Little Landmarks},

Computer Vision and Pattern Recognition,

2016.

\bibitem{Zisserman2015}

Andrew Zisserman, Karen Simonyan,

\textit{Very Deep Convolutional Networks for Large-Scale Image Recognition},

Computer Vision and Pattern Recognition,

2015.

\bibitem{Russakovsky2015}

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei,

\textit{ImageNet Large Scale Visual Recognition Challenge},

Computer Vision and Pattern Recognition,

2015.

\bibitem{Fergus2013}

Rob Fergus, Matthew D Zeiler,

\textit{Visualizing and Understanding Convolutional Networks},

Computer Vision and Pattern Recognition,

2013.

\bibitem{Fergus2011}

Rob Fergus, Matthew D Zeiler,

\textit{Adaptive deconvolutional networks for mid and high level feature learning},

International Conference on Computer Vision,

2011.

\bibitem{Chen2017}

Chen C., Liu MY., Tuzel O., Xiao J.,

\textit{R-CNN for Small Object Detection},

Asian Conference on Computer Vision,

2017.

\bibitem{Ren2016}

Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun,

\textit{Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks},

Computer Vision and Pattern Recognition,

2016.

\bibitem{Eggert2017}

Christian Eggert, Dan Zecha, Stephan Brehm, Rainer Lienhart,

\textit{Improving Small Object Proposals for Company Logo Detection},

Computer Vision and Pattern Recognition,

2017.

\bibitem{Zuo2015}

Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang,Yushi Chen,

\textit{Convolutional recurrent neural networks: Learning spatial dependencies for image representation},

Conference on Computer Vision and Pattern Recognition Workshops,

2015.

\bibitem{Pan2009}

Sinno Jialin Pan and Qiang Yang.

\textit{A Survey on Transfer Learning},

IEEE Transactions on Knowledge and Data Engineering,

2009.

\bibitem{Shin2016}

Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, Ronald M. Summers,

\textit{Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning},

Computer Vision and Pattern Recognition,

2016.

\bibitem{Hastie2014}

Trevor Hastie, Gareth James, Daniela Witten, Robert Tibshirani,

\textit{An Introduction to Statistical Learning},

Springer,

2014.

\bibitem{Goodfellow2016}

Ian Goodfellow, Yoshua Bengio, Aaron Courville,

\textit{Deep Learning},

MIT Press,

2016.

\bibitem{Srivastava2014}

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov,

\textit{Dropout: A Simple Way to Prevent Neural Networks from Overfitting},

Journal of Machine Learning Research,

2014.

\bibitem{SKrause2013}

Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei,

\textit{Object Representations for Fine-Grained Categorization},

4th International IEEE Workshop on 3D Representation and Recognition,

2013.

\bibitem{Uijlings2013}

J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders,

\textit{Selective Search for Object Recognition},

International Journal of Computer Vision,

2013.

\bibitem{Godil2014}

Afzal A. Godil, Roger V. Bostelman, William P. Shackleford, Tsai Hong Hong, Michael O. Shneier

\textit{Performance Metrics for Evaluating Object and Human Detection and Tracking Systems },

NIST Interagency/Internal Report,

2014.

\bibitem{Bengio2017}

Yoshua Bengio,

\textit{Deep Learning of Representations for Unsupervised and Transfer Learning},

Proceedings of Machine Learning Research,

2017.

\bibitem{Oquab2014}

Maxime Oquab, Leon Bottou, Ivan Laptev1, Josef Sivic1,

\textit{Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks},

Computer Vision and Pattern Recognition,

2014.

\bibitem{Quinn2017}

Max H. Quinn, Erik Conser, Jordan M. Witte, Melanie Mitchell,

\textit{Semantic Image Retrieval via Active Grounding of Visual Situations},

Computer Vision and Pattern Recognition,

2017.

\bibitem{machrisaa2017}

machrisaa,

\textit{VGG-16 TensorFlow},

https://github.com/machrisaa/tensorflow-vgg,

2017.

\bibitem{Ahmed2008}

Amr Ahmed, Kai Yu, Wei Xu, Yihong Gong, Eric Xing,

\textit{Training Hierarchical Feed-Forward Visual Recognition Models Using Transfer Learning from Pseudo-Tasks},

Springer,

2008.

\bibitem{Collobert2011}

Ronan Collobert,Jason Weston,

\textit{Natural Language Processing (Almost) from Scratch},

Journal of Machine Learning Research,

2011.

\bibitem{Xu2016}

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio,

\textit{Show, Attend and Tell: Neural Image Caption Generation with Visual Attention},

Computer Vision and Pattern Recognition,

2016.

\bibitem{Erhan2010}

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, Samy Bengio,

\textit{Why does Unsupervised Pre-training Help Deep Learning?},

Journal of Machine Learning Research,

2010.

\bibitem{Krizhevsky2012}

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton,

\textit{ImageNet Classification with Deep Convolutional Neural Networks},

NIPS,

2012.

\bibitem{Murphy2012}

Kevin Murphy,

\textit{Machine Learning},

MIT Press,

2012.

\end{thebibliography}

\end{document}