What's the story

Concept Learning - understanding the dynamics of relationship between objects - like person, dog and leash. This is

done by attempting to localize on of hand-holding leash and attached-to-leash part of the image.

What are the challenges in localizing on hand-holding-leash - the small object localization problem.

Learning by use of context. Use other examples.

**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 - like location of hand-holding-leash - in a dog walking image, we create a regression model using Convolutional Neural Network (CNN) that is supervised by the coordinates of the small object in an 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 in [1] for small object localization by using a multistep regression process where spatial context is used effectively to localize on door handle of a car 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 reduce the training time.

**Introduction**

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 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. [1] proposed the idea of using successively more relevant contextual information in a sequence of steps using a recurrent 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 other 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® we implement a model of the CNN model described in [1]. We use the car door handle dataset, provided by authors of [1], 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 xxx. In section xxx 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 achieves 92.7% accuracy in top-5 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 xxx.

**Related work**

Localization of small object in an image using CNN is little understudied. Chen 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 and 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 xxx.

Another important work for small object identification is done by Eggert et al. in [8] where they modify Faster R-CNN 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 VGG16. 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 encode 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 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 result in significant improvement in training time. Pan 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 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) 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.

Attention based models

**Deep Learning/CNN**

Since its resurgence in 2012 when a neural network based architecture, called AlexNet [Krizhevsky et al.], was proposed 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 by [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).

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 below. 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. 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 [Murphy 2012 (Machine Learning, section 16.5.1)].

|  |  |
| --- | --- |
| nput  24x24  Convolution  Feature maps  4@20x20  Feature maps  4@10x10  Subsampltng  Feature maps  8@8x8  Convolution  Feature maps  8@4x4  Subsampling  Output  20@1x1  Convolution | Macintosh HD:Users:sharadkumar:Library:Group Containers:UBF8T346G9.Office:msoclip1:01:A8541945-4B93-EE4C-B5DD-FA20FB0580B0.png |
| Figure 1: Explain    *From:* [*https://www.slideshare.net/hanneshapke/introduction-to-convolutional-neural-networks*](https://www.slideshare.net/hanneshapke/introduction-to-convolutional-neural-networks) | Figure 2: Illustrates filter's transformation of the input into output. The figure illustrates two-dimensional input, but usually the input also has a third dimension of depth which is matched by filter's third dimension |

***Forward Pass***

*Briefly explain about how forward pass is used in little landmark model*

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 [Krizhevsky, ImageNet Classification with Deep Convolutional Neural Networks], 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 [Murphy 2012 (Machine Learning, section 16.5.1), Hastie et al., Essentials of Statistical Learning (Section 11.3)]. Neural networks can therefore be thought of as a nonlinear generalization of the linear model and by introducing the nonlinear transformation it greatly enlarges the class of linear model [Hastie et al., Essentials of Statistical Learning (Section 11.3)].

Activations from the ReLU layer are subsampled by the pooling layer. As shown in Figure3, 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.

|  |
| --- |
| 5 9  12 7  Average Pooling  •aan  21 12  18 10  Max Pooling |
| Figure 3: Explain    *From:* <https://medium.com/@Aj.Cheng/convolutional-neural-network-d9f69e473feb> |

***Back Propagation***

*Briefly explain about how backprop is used in little landmark model*

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 test dataset. 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 and chain rule of derivatives are used to update model weights iteratively in minibatches. Use of minibatches in stochastic GD 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 [Goodfellow et al. (Deep Learning, 2016, Section 8.3)].

***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 [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 [13]. L1 and L2 are two most commonly used regularization techniques in machine learning in general, including neural networks. Dropout [14] is another regularization technique used specifically with neural network. In our implementation of localizing on dog walking image, we explored L1 as well as L2 regularization as well as dropout and we report our findings in the experiments section xxx. In [1] authors use L2 regularization as explained in section xxx.

***Classification vs. Regression?***

**Main Paper**

***Architecture Overview***

An efficient object detection model in an image may use region proposals, as one used by Girshick *et al.* in R-CNN [7, 15]. This 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 et al. [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 *et al.* [6], low resolution inputs for small objects is 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 et al. [1] propose a stepwise regression model for this task and they call this model an architecture for localizing *little landmark*. This is a recurrent model that is trained end to end using the location of the little landmark. Authors demonstrate the robustness of this model by testing and training it on several datasets. One of contribution of their work [1] was to create and annotate a dataset of little landmarks and for this task authors use Stanford's car dataset [15] 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 xxx.

This CNN architecture exploits the fact that a little landmark is defined by it's context. Authors define *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 below 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.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | Macintosh HD:Users:sharadkumar:Library:Group Containers:UBF8T346G9.Office:msoclip1:01:5D77522E-4038-7244-B1A4-7749081A0C08.png | Macintosh HD:Users:sharadkumar:Library:Group Containers:UBF8T346G9.Office:msoclip1:01:812F659C-E037-A646-8FF4-EC01FDFB72C8.png | |
| Figure 4: explain |

Figure 5 below shows a schematic diagram of the little landmark localization architecture that localizes on a car door handle in three steps.

|  |
| --- |
| Z tpod  "9 'Z 'E 'ElZP0d  't 'st •st)tnuo,  (OS 't 'S •slEnuo' |
| Figure 5: explain |

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, *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 is explained later in section xxx, but it's worth noting here that the difference between prediction (blue blob) in step *s-1* and the generated latent landmark location (red blob) in step *s* contributes to total error of an iteration.

Weights in the bottom portion and in row two and three of *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

***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, *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 section xxx.

***Latent Landmark Prediction***

Location of target little landmark is only supervision used in the model and the 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. This process is explained next.

*Latent Landmark Centroid Generation (red blob in figure 5)*

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

*pc(s) = Σl qi(s) \* out\_locsl* (1)

where *qi(s)*is softmax over all locations computed as *qi(s)*=*ezi(s)/Σi ezi(s)* and *zi(s)*is the output from network for confidence at *li*

*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 *li* is used for generating *P(s)*. **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 *5 x 5* cells is placed over each location *li*. Last 25 layers of *w* x *h* x *26* dimensional output from CONV/RELU11 is used for this calculation. This scheme is illustrated in Figure 6 below.

|  |
| --- |
| 【 00 6 「 を PO イ  【 oc 芝 0 コ dCon ま ス  OffsetP 希 d 一 & 0 コ  一 ョ 品 の 2 P d を ユ 03  ー 3 6 ま 9 コ 諡 de コ 日  P 希 d 一 & 0 コ fo 「 Step  Predicted 【 oc 当 を コ ま 「 コ t  「 & ョ 当 k |
| Figure 6: explain |

Left hand side of Figure 6 shows 25 logical grid points *gj* corresponding to a locations *li* (as shown in right side of Figure 6), where *gj(x), gj(y)* ε{-50, -25, 0, 25, 50} pixels. The network produces 25 confidence values *oji(s)* for j ε {1, …, 25}from last 25 layers of output from CONV/RELU11 layer. These *oji(s)* are a softmax of network outputs.

Each location *li* produces the estimate of next latent landmark as follows:

*pi(s) =* Σ*j oji(s)gj* (2)

These *pi(s)* values are used with confidence values *qi(s)*, produced by first layer of the output from CONV/RELU11 layer, to produce *poc* as follows:

*poc(s) = Σl qi(s) \* pi(s)* (3)

Finally, to generate prediction *P(s)* for latent landmark of the next step, encoding is done by placing a radial basis kernel with **β** = 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.

***Training (and Loss Calculation)***

We train the model through back-propagation using Adam optimizer. *L2* loss between predicted location and the ground truth is used for gradient calculation. However, there are several components in loss value. *L2* loss between the ground truth and the location predicted by the last step of the model and the *L2* loss between latent landmark predicted in step *s* and estimated in step *s-1* contribute to the total loss of the model in one iteration, which is averaged over a 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 *L2* loss between ground truth and predicted location in the last step as shown below

Macintosh HD:Users:sharadkumar:Library:Group Containers:UBF8T346G9.Office:msoclip1:01:203725E4-8621-6E42-A475-A24743083D4C.png

We use **λ**s= 0.1, except for the final step S where **λ**s = 1. *R*(**θ**) is a regularizer for the parameters of the network. We use *L2* 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 (as explained in section xxx) 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.

Regularization?

**Performance Metric**

To evaluate model accuracy, we use a generally accepted [17] metric of plotting detection rate against normalized distance from ground truth. The *L2* 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.

**Dataset**

*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.

*Dog Walking Images (DWI) Dataset*

We used our own dog walking images to test model's effectiveness on a different dataset. We had 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.

**Tensorflow Implementation of the Model**

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 and Python. We tested fidelity of our implementation by comparing our test results with the results authors published in their paper [1]. We used as same dataset and for training and testing the model and found that out model performed similar to authors'. Figure 7 shows a comparison between the models.

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.

*Training Detail*

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

|  |  |
| --- | --- |
| .4  02  0.02  0.04  Img  0.0B  Normalized Distance —9 | Macintosh HD:Users:sharadkumar:Library:Group Containers:UBF8T346G9.Office:msoclip1:01:3E203A8E-CC6B-B341-BD30-DFD923F23C67.png |
| Figure 7(a): explain | Figure 7(b) |

**Models Modifications and Experiments**

As explained before, the model proposed in [1] takes more than 100 hours to train. Our proposal to improve training time for this model involves using transfer learning where we use activations from another state-of the-art model pre-trained on an auxiliary dataset as the starting point of our model training. We used transfer learning for training the model and observed significant improvement in training time. We tested the new model trained on car door handle dataset and saw comparable model accuracy. We also trained and tested this new model on our dog walking images. We report our findings in section xxx.

The rational behind TL is xxx

Discuss what transfer learning is

CDHD is trained end to end, all the layers are initialized with a random normal distribution and all the layers are trained from scratch

Different layers of the network learn different types of abstractions (explain more). Murphy pp 995 - explain transfer learning using this

Discuss what VGG-16 model is and how was transfer learning incorporated in the model

Explain bridge layer

Discuss Results

It took 27 hours to achieve same result with transfer learning

Training the entire model vs. fine tuning

Explain two models

First for localizing hand-holding-leash

Second for extending the model for identifying two small objects in one training

Explain the vanishing gradient problem that happened in getNormalizedLocationWeightsFast() function and how it was solved by using normalization. Overlapping Pooling may have solved this problem (ImageNet Classification with Deep Convolutional Neural Networks, section 3.4)

Pooling layer is not used since downsampling by maxpooling results in low resolution feature map which may make it difficult to identify small image. (logo paper)

Dataset

CDHD and Dog Walking images

Explain the problem of over-fitting

that it may be because the context around hand holding leash is not sufficient for localizing

It may be because of not enough data and transfer learning did not help in this case

Use pictures to show where latent coordinates and the final coordinates fall during training, which is pretty close

Discussed ideas used to solve the over-fitting - different hyper-parameters (momentum, weight decay and dropouts)

dropout also has the amazing property that it prevents co-adaptation of feature detectors which improves a network’s ability to generalize (<https://dmm613.wordpress.com/2014/10/15/intriguing-properties-of-neural-networks/>)

Discuss about use of different types of optimizers - Adam and SGD

In this case fine tuning is done to the columns. The assumption is that VGG 16 model is generalized enough and does not have overfitting problem. Because of this the weights in the VGG-16 model are not changed.

**Conclusion:**

One of the reasons why DWI did not localize:

[1] The target landmark may have a local appearance that is similar to many other locations in the image. However, it may occur in a consistent spatial configuration with some pattern, such as an object or part, that is easier to find and would resolve the ambiguity.

There are much more possibility of the location of small objects (RCNN for small objects)

Metric used for measuring performance of small object detection is not consistent - some researchers use IoU whereas others use some sort of normalization. A future work for this is developing good metric for measuring performance of small object detection.

The spatial configuration around the car door handle and an electrical switch is fairly uniform across images, but the same is not true for dog walking images.

Future Work:

1. Unsupervised Learning - The idea here is to initialize the weights using pre-training so that the weights only change a "little bit" during backprop (back propagation in this case is called fine tuning). This kind of pre-training is like regularizaton since it trying to reduce overfitting by reducive variance in the model.

Use Restricted Boltzman Machones or autoencoders or denoising autoencoders for pre-traing to initialize weights

<https://www.youtube.com/watch?v=Oq38pINmddk> (Hugo Larochelle)

Paper: Why Does Unsupervised Pre-training Help Deep Learning (section 3) - greedy layer-wise unsupervised pre-training overcomes the challenges of deep learning by introducing a useful prior to the supervised fine-tuning training procedure

<https://metacademy.org/graphs/concepts/unsupervised_pre_training>

Learning Invariant Feature Hierarchies - Yann LeCun (<http://yann.lecun.com/exdb/publis/pdf/lecun-eccv-12.pdf>)

1. Attention Model [Multiple object recog- nition with visual attention]
2. Shortcomings of the metric of normalizing by height of the bounding box, what other options are
3. Explore possibility of removing pooling layers from pre-trained VGG model. This may help since downsampling in the pooling layer renders the low resolution feature map making it difficult to identify a small object (logo paper, page 1, Introdiution section)
4. R-CNN for small objects - Mitsubishi Labs
5. Visualization using t-SNE
6. Metric used for measuring performance of small object detection is not consistent - some researchers use IoU whereas others use some sort of normalization. A future work for this is developing good metric for measuring performance of small object detection. R-CNN for small object paper mentions that it's difficult to quantify how hard this problem is (since there is no good metric for measuring accuracy)

Bibliography

[1] Singh, Hoiem and Forsyth. Learning to Localize Little Landmarks. In CVPR 2016.

[2] VGG-16 model

[3] Imagenet dataset

[4] Visualizing and Understanding Convolutional Networks - Fergus

[5] Adaptive deconvolutional networks for mid and high level feature learning

[6] R-CNN for Small Object Detection, Chen, C.; Liu, M.-Y.; Tuzel, C.O.; Xiao, J.

[7] Faster R-CNN

[8] Company Logo Detection

[9] Convolutional Recurrent Neural Networks: Learning Spatial Dependencies for Image Representation

[10] A Survey on Transfer Learning

[11] Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning

[12] ISLR, Hastie

[13] Deep Learning book, chapter 7

[14] Dropout, Hinton

[15] <https://ai.stanford.edu/~jkrause/cars/car_dataset.html>

[16] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. IJCV, 2013

[17] Performance Metrics for Evaluating Object and Human Detection and Tracking Systems <https://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7972.pdf>

Open questions:

ROC curves for regression (José Hernández-Orallo)

How do we know that the training is not stuck in local minima