Comp 766 Assignment 2

Jonathan Pearce 260672004

July 23, 2020

Problem 1.

Likelihood Function

$$L(\mathcal{M} : \mathcal{D}) = P(\mathcal{M}|\mathcal{D}) = \prod_{m=1}^{M} P(x[m]|\mathcal{T}, A)$$

Here $\mathcal{M} = (\mathcal{C}, \mathcal{T}, \mathcal{A})$ is the module network. $\mathcal{C}$ is the set of modules $\{M_1, ..., M_K\}$. $\mathcal{T}$ = (S,θ) is a module network template for $\mathcal{C}$, where S denotes the dependency structure encoded by $\{Pa_{M_j} : M_j \in \mathcal{C}\}$ and θ denotes the parameters required for the conditional probability template $\{P(M_j|Pa_{M_j}) : M_j \in \mathcal{C}\}$. $\mathcal{A}$ is a module assignment function for $\mathcal{C}$, with $\mathcal{A}(X_i) = j$ if Val(X_i) = Val(M_j). $\mathcal{A}$ is a function that is equal for any two random variables in the same module and is otherwise not equal. An important detail that will be crucial in the discussion of the learning algorithm is that a module network $\mathcal{M}$ induces a module graph $\mathcal{G}_{\mathcal{M}}$ that is acyclic. $\mathcal{D}$ = {x[1],.., x[M] }, is a training set consisting of M instances drawn independently from an unknown distribution P($\mathcal{X}$)

In the paper the authors discuss how this global likelihood function can be decomposed and considered as a product of local module likelihoods across all K modules.

$$L(\mathcal{M} : \mathcal{D}) = \prod_{j=1}^{K} L_j(Pa_{M_j}, X^j, \theta_{M_j|Pa_{M_J}} : \mathcal{D})$$

Here $X_j = \{\text{X} \in \mathcal{X} \mid \mathcal{A}(\text{X}) = j\}$, Pa_{M_j} are the parents of module j and $\theta_{M_j|Pa_{M_J}}$ are the parameters associated with the conditional probability distribution $P(M_j \mid Pa_{M_J})$. The key point is that the module likelihood of module j can be calculated independently from all other modules and depends only on X_j, Pa_{M_J} and $\theta_{M_j|Pa_{M_J}}$.

They provide formulation for L_j

$$L_j(Pa_{M_j}, X^j, \theta_{M_j|Pa_{M_J}} : \mathcal{D}) = \prod_{x,u \in Val(M_j, Pa_{M_j})} \theta_{x|u}^{\hat{S}_j[x,u]}$$

Here $\theta_{x|u}^{\hat{S}_j[x,u]}$ are sufficient statistics calculated based on pooling the data across all the variables in X^j. This decomposition of the likelihood function is key in allowing us to perform maximum likelihood or MAP estimation efficiently, optimizing the parameters for each module independently

Priors and the Bayesian Score

The authors introduce a Bayesian model score for a pair ($\mathcal{S}$,$\mathcal{A}$) as the posterior probability of the pair, integrating over all choices of parameters θ. They then define an assignment prior P($\mathcal{A}$), a structure prior P($\mathcal{S} \mid \mathcal{A}$) and a parameter prior P($\theta \mid \mathcal{S}, \mathcal{A}$). As stated by the authors, these describe the preferences over different networks before seeing the data $\mathcal{D}$. Leveraging Bayes' rule to expand P($\mathcal{S}, \mathcal{A} \mid \mathcal{D}$)

$$P(\mathcal{S}, \mathcal{A} \mid \mathcal{D}) \propto P(\mathcal{A})P(\mathcal{S} \mid \mathcal{A})P(\mathcal{D} \mid \mathcal{S}, \mathcal{A})$$

They define a Bayesian score as the log of P($\mathcal{S}$,$\mathcal{A}$ | $\mathcal{D}$), ignoring the normalization constant.

$$score(\mathcal{S}, \mathcal{A} : \mathcal{D}) = logP(\mathcal{A}) + logP(\mathcal{S}|\mathcal{A}) + logP(\mathcal{D}|\mathcal{S}, \mathcal{A})$$

Next the authors consider some conditions that ensure that module network learning can be done efficiently, Bayesian network learning requires very similar conditions to be met for fast training. These conditions, when met, allow for the global Bayesian score to be decomposed into local module scores (similarly to what we did with the likelihood function)

$$score(\mathcal{S}, \mathcal{A} : \mathcal{D}) = \sum_{j=1}^{K} score_{M_j}(Pa_{M_j}, A(X^j) : \mathcal{D})$$

They provide an explicit decomposition of the local module score,

$$score_{M_j}(U, X : \mathcal{D}) = log \int L_j(U, X, \theta_{M_j|U} : \mathcal{D}) P(\theta_{M_j} \mid \mathcal{S}_j = U) + logP(\mathcal{S}_j = U) + logP(\mathcal{A}_j = X)$$

Here we choose a structure where U are the parents of the module M_j, and $\mathcal{A}_j = X$ denotes that $\mathcal{A}$ is such that $X^j = X$. This decomposition is very important in ensuring we are able to efficiently search the module network space (via the learning algorithm) for a module network with a structure that induces a high score.

Learning Algorithm
We now shift the focus towards the learning algorithm outlined in the paper. The goal of the learning algorithm is find a high scoring module network. This search is completed over two combinatorial spaces, the space of network structures $\mathcal{S}$ and the space of module assignments $\mathcal{A}$. This difficulty of searching two spaces is what motivates their alternating optimization algorithm. Their iterative approach is to fix the module assignments and optimize the module network structure (structure search step), then they proceed to flip that process and fix the module network structure and optimize the module assignments (module assignment search step).

Structure Search Step
In this part of the iterative learning algorithm, we search for an optimal structure $\mathcal{S}$ with a fixed module assignment $\mathcal{A}$. The goal is to find a structure $\mathcal{S}$ that maximizes $score(\mathcal{S}, \mathcal{A} : \mathcal{D})$. The authors discuss that this process is the same as structure learning in Bayesian networks and they use a heuristic search over the module network space. The authors give an example of a common search operator that can be used in this search. The operator involves adding or removing a variable X_i from a parent set Pa_{M_j} while ensuring that the module graph $\mathcal{G}_{\mathcal{M}}$ remains acyclic and thus the module network is valid. Testing that the module network is acyclic is a good example of where this modular framework can have huge computational advantages over standard Bayesian networks. Cyclicity tests only need to consider the K nodes of the module graph, rather than the n nodes of a Bayesian network dependency graph, noting that usually n is much larger than K. Using an operator such as the one described above one can search through the space of module network structures in search of high scoring structures using a standard search algorithm, the authors provide greedy hill climbing as an example. Finally the authors include a quick discussion focusing on the fact that decomposing the global scoring term into local module scoring terms provides more computational savings in that after applying an operator to Pa_{M_j}, one only needs to update the score for those operators that involve M_j

Module Assignment Search Step
The second part of the iterative learning algorithm is to search for an optimal module assignment $\mathcal{A}$ with a fixed structure $\mathcal{S}$.

$$\mathcal{A} = argmax_{\mathcal{A}'}\ score_M(\mathcal{S}, \mathcal{A}' : \mathcal{D})$$

The log marginal likelihood is not additive in the sufficient statistics of the different variables, therefore we are limited to only be able to compute the change in score for moving a variable from one module to another given a fixed assignment of the other variables in the two modules. Therefore the authors propose a sequential update algorithm that reassigns the variables to modules one by one. The algorithm is simple, given an initial assignment, iterate over all variables one by one. For each variable X_i keep all other variables

fixed, now we find the optimal assignment for X_i while keeping the module graph acyclic. Continue this process until no reassignment can improve the score. This sequential algorithm ensures that we reach a local maximum. The decomposition of the score into local module scores also has a significant benefit that is discussed, when reassigning a variable X_i from M_j to M_k, only the local score of these modules changes, and therefore needs to be recomputed.

Initialization
This iterative learning algorithm requires an initial module network assignment. Since the goal of these module networks is to group variables that behave similarly the authors equate it to a clustering problem, where the variables are the instances to be clustered. The authors describe their initialization approach as a model merging approach. To start their initialization they introduce a dummy variable U that encodes training instance identity $u[m] = m$ for all m. Next they create n modules with $\mathcal{A}(X_i) = i$ and $Pa_{m_i} = U$. Observe that each instance and each variable has its own local probabilistic model. Next they introduce the model merging, they consider all possible legal module mergers where the assignment function is changed such that modules j_1 and j_2 become module $j_{1,2}$, now each instance shares parameters. We greedily select the merger that leads to the best scoring network, and repeat this procedure until we reach a module network structure with a target number of modules.

Convergence
The iterative algorithm begins with an initialization as outlined above. The 2 steps of the iterative algorithm are designed to either improve the score or leave it unchanged. Therefore the iterative module network learning algorithm converges to a local maximum of $score(\mathcal{S}, \mathcal{A} : \mathcal{D})$.

Learning with Regression Trees
Before showing their experimental results the authors quickly discuss that in order for their framework to work with continuous valued variables, they need to represent these conditional probability models as regression trees. They then go over the adaptations required to allow their iterative learning algorithm and scoring function to work with regression trees.

Problem 2.

CNN forward pass

The forward pass of a CNN as stated in the paper is as follow,

$$f(X, \{W_i\}_{i=1}^2 \{b_i\}_{i=1}^2) = Z_2 = \text{ReLU}(W_2^T \text{ ReLU}(W_1^T X + b_1) + b_2)$$

Here $X \in \mathbb{R}^N$ is the one dimensional input. $W_1 \in \mathbb{R}^{N \text{x} N m_1}$ is a matrix whose columns contain m_1 learned filters of length n_0 with all of their shifts. W_1 is denoted a covolutional matrix with stride 1, meaning the filters are offset by 1 position. $b_1 \in \mathbb{R}^{N m_1}$ is a bias term. Similarly $W_2 \in \mathbb{R}^{N m_1 \text{x} N m_2}$ is a convolutional matrix with m_2 filters of length $n_1 m_1$, W_2 has stride m_1 (filters offset by m_1 positions). $b_2 \in \mathbb{R}^{N m_2}$ is also a bias term.

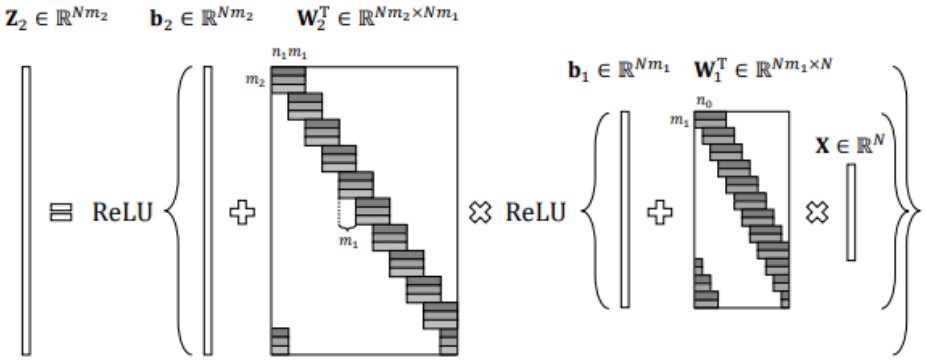

Figure 1: CNN forward pass with one dimensional input X

ReLU is the rectified linear unit with formula,

$$\text{Relu}(x) = max(x, 0)$$

The matrix vector multiplication $W_1^T X$ represents the convolution of the input X with the m_1 filters in W_1, this computation creates m_1 feature maps. The bias term b_1 is added and then the rectified linear unit is applied to this vector. The result of this computation is $Z_1 \in \mathbb{R}^{N m_1}$, explicitly,

$$Z_1 = \text{ReLU}(W_1^T X + b_1)$$

Then the second convolution involving W_2 and b_2 can be represented as,

$$Z_2 = \text{ReLU}(W_2^T Z_1 + b_2)$$

We observe that this convolution takes the same form as the first convolution, with X replaced by Z_1.

Sparse coding

The goal of sparse coding is to take a signal $X \in \mathbb{R}^N$ and given a fixed dictionary $D \in \mathbb{R}^{N \text{x} M}$, find a vector $\Gamma \in \mathbb{R}^M$ such that the matrix vector product $D\Gamma$ is equal to X and Γ is as sparse as possible. More formally,

$$\min_{\Gamma} \|\Gamma\|_p \quad s.t. \quad D\Gamma = X$$

Where p is usually 0 or 1 since the 0 and 1-norm naturally induce sparsity. Usually an exact solution is not possible and so algorithms seek optimal solutions for Γ that minimize some error function. Typically these minimization problems are expressed as follows,

$$\min_{\Gamma} \frac{1}{2} \|\Gamma - D^T X\|_2^2 + \beta \|\Gamma\|_p$$

In the paper they discuss that this formulation above is a projection problem and has a closed form solution in the form of $\mathcal{H}_\beta(D^T X)$ or $\mathcal{S}_\beta(D^T X)$. Where $\mathcal{H}_\beta(z)$ and $\mathcal{S}_\beta(z)$ are sparsifying operators. For the purposes of this question we will focus on $\mathcal{S}_\beta(z)$, which has form,

$$\mathcal{S}_\beta(z) = \begin{cases} z+\beta, & \text{if } z < -\beta \\ 0, & \text{if } -\beta \leq z \leq \beta \\ z-\beta, & \text{if } \beta < z \end{cases}$$

The key is that $\mathcal{S}_\beta(z)$ promotes a sparse solution. The authors of the paper then show how if we enforce the entries of the sparse vector Γ to be nonnegative, this does not change its expressiveness. In otherwords they are able to reduce the sparse coding problem to the nonnegative sparse coding problem. A direct consequence of this is that the projection problem described above now is constraint to nonnegative values and thus has a solution of the form $\mathcal{S}^+_\beta(D^T X)$, where $\mathcal{S}^+_\beta(z)$ is of the form,

$$\mathcal{S}^+_\beta(z) = \begin{cases} 0, & \text{if } z \leq \beta \\ z-\beta, & \text{if } \beta < z \end{cases}$$

It can be seen that this equation admits another formulation,

$$\mathcal{S}^+_\beta(z) = max(z-\beta, 0) = \text{ReLU}(z-\beta)$$

This relation proves very important in relating the CNN forward pass to convolutional sparse coding.

Multi-Layer sparse coding
Going back and using the exact solution for sparse coding we have, the authors discuss the possibility of decomposing D into 2 matrices D_1 and D_2. Where D_1 is a dictionary with fast implementation and D_2 is a trained sparse dictionary matrix. Now we have,

$$X = D\Gamma_2 = D_1 D_2 \Gamma_2$$

The authors postulate that since D_2 and Γ_2 are both sparse it is likely there multiplication (denoted $\Gamma_1 = D_2\Gamma_2$) is sparse as well. Therefore we can consider this sparse coding problem as a 2 stage problem. First decompose input X into a product of dictionary D_1 and sparse vector Γ_1. From there we can decompose Γ_1 into D_2 and Γ_2 in a similar manner.

Convolutional sparse coding
Usually the sparse representation model is used for modelling local patches of a global signal (e.g. a $n \times m$ patch of pixels from an image). The convolutional sparse coding model is a problem that tries to decompose the entire global signal X. X can be decomposed into a global convolutional dictionary $D \in \mathbb{R}^{N \times Nm}$ and sparse vector $\Gamma \in \mathbb{R}^{Nm}$. Where D is created by shifting a local matrix of size $n \times m$ in all possible positions. Refer to D_1 in Figure 2 below, where $n = n_0$ and $m = m_1$

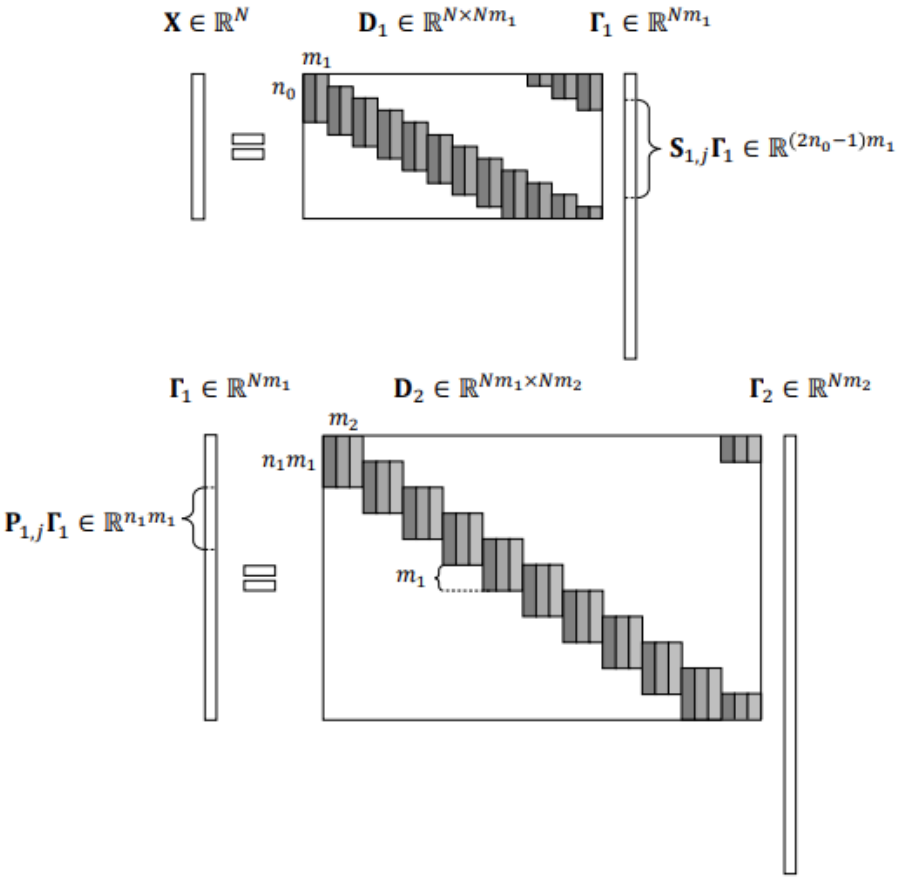

Figure 2: Multi-layer convolutional sparse model with one dimensional input X

Multi-Layer Convolutional Sparse Model

From above we know that a convolutional sparse model attempts to decompose (with minimal signal loss) a global signal $X \in \mathbb{R}^N$ into a multiplication of a convolutional dictionary $D_1 \in \mathbb{R}^{N \text{x} Nm_1}$ with stride 1, composed of m_1 local filters on length n_0 and a sparse vector $\Gamma_1 \in \mathbb{R}^{Nm_1}$. If we now consider Γ_1 as a new global signal this process can be repeated, with $D_2 \in \mathbb{R}^{Nm_1 \text{x} Nm_2}$ with stride m_1, composed of m_2 local filters on length $n_1 m_1$ and a sparse vector $\Gamma_2 \in \mathbb{R}^{Nm_2}$. This can be expressed formally using 2 equations:

$$X = D_1\Gamma_1$$

$$\Gamma_1 = D_2\Gamma_2$$

Therefore our goal would be to find Γ_2 given X,D_1,D_2. Very early on in this discussion we addressed the fact that reaching equality in these equations is nearly impossible and that these sparse coding problems are better formulated as minimization problems that seek to find solutions for Γ that with a fixed dictionary D minimize the loss in signal in recreating X and ensure Γ is sparse. This was followed by the fact that these minimization problems were just projection problems and had solutions of the form $\mathcal{P}_\beta(D^T X)$ where $\mathcal{P}_\beta(z)$ is a sparsifying operator. Therefore our goal of finding an optimal sparse vector Γ_2 is best expressed as follows, where $\hat{\Gamma}_2$ is an approximation of Γ_2:

$$\Gamma_2 \approx \hat{\Gamma}_2 = \mathcal{P}_{\beta_2}(D_2^T \hat{\Gamma}_1)$$

Similarly the approximation of Γ_1, $\hat{\Gamma}_1$ can be expressed

$$\Gamma_1 \approx \hat{\Gamma}_1 = \mathcal{P}_{\beta_1}(D_1^T X)$$

If we combine these two equations the result is as follows,

$$\Gamma_2 \approx \hat{\Gamma}_2 = \mathcal{P}_{\beta_2}(D_2^T \mathcal{P}_{\beta_1}(D_1^T X))$$

Recalling the formula for a forward pass in a CNN the two expressions are quite similar.

$$f(X, \{W_i\}_{i=1}^2 \{b_i\}_{i=1}^2) = \text{ReLU}(W_2^T \text{ ReLU}(W_1^T X + b_1) + b_2)$$

In fact if we let $\mathcal{P}_\beta(z) = \mathcal{S}_\beta^+(z)$, then we have,

$$\hat{\Gamma}_2 = \mathcal{S}_{\beta_2}^+(D_2^T \mathcal{S}_{\beta_1}^+(D_1^T X))$$

$$\Rightarrow \hat{\Gamma}_2 = \mathcal{S}_0^+(D_2^T \mathcal{S}_0^+(D_1^T X - \boldsymbol{\beta_1}) - \boldsymbol{\beta_2})$$

Where $\boldsymbol{\beta_1} = (\beta_1, ..., \beta_1)^T \in \mathbb{R}^{Nm_1}$ and $\boldsymbol{\beta_2} = (\beta_2, ..., \beta_2)^T \in \mathbb{R}^{Nm_2}$. Now recall $\mathcal{S}_0^+(z) = \text{ReLU}(z)$

$$\Rightarrow \hat{\Gamma}_2 = \text{ReLU}(D_2^T \text{ ReLU}(D_1^T X - \boldsymbol{\beta_1}) - \boldsymbol{\beta_2})$$

This equivalent formulation of multi-layer sparse coding is virtually identical to the forward pass equation of a CNN. The only difference being that the bias vectors in the multi-layer sparse coding formulation $\boldsymbol{\beta_1}$ and $\boldsymbol{\beta_2}$ are constant vectors, where as b_1 and b_2, the bias vectors in the CNN forward pass are not required to be constant.

Problem 3.
I attempted to get U-Net working on my laptop (MacBook running OSX), I was unable to resolve errors pertaining to the mex file and protobuf 8. I faced similar issues with DeConvNet. An issue with Caffe installation or my MatLab software was most likely the reason for the errors, however I was unable to resolve them in the time I had available. I also attempted to get U-Net working on a computer on the 3rd floor of the Trottier building, however I had problems setting up caffe properly in the limited time I had left at that point. Therefore I have instead used a python based segmentation tool from `https://github.com/divamgupta/image-segmentation-keras`. I found this repository, which contains a pretrained Pyramid Scene Parsing Network (PSPNet) model [1]. Elliot had found the same repository and shared the segmentations it produced with me. For this question I will give a brief overview of what PSPNet is, discuss the results that PSPNet produced, discuss the results of segmentation models trained on the PASCAL VOC 12 dataset and discuss why U-Net's segmentation would have most likely been better.

PSPNet
The pretrained PSPNet that I used to obtain image segmenations was originally trained on the PASCAL VOC 2012 dataset. PSPNet obtained a record of mean IoU accuracy 85.4% on this dataset, for reference DeconvNet had a mean IoU accuracy of 69.6% on the same dataset, therefore I was confident that at the very least PSPNet would provide competitive segmentation results and allow for a useful discussion.

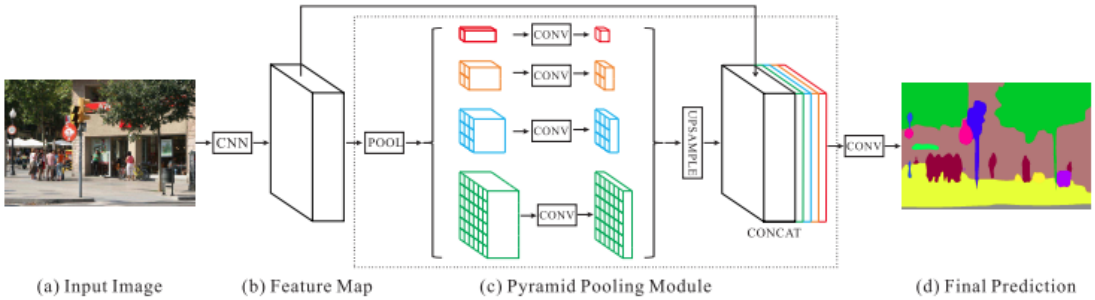

Figure 3: PSPNet architecture

PSPNet was designed for semantic image segmentation, where each pixel in an image is assigned a class label. The input image is initially fed through a pretrained ResNet model to extract a feature map, the feature map is 1/8th the size of the original image. Next PSPNet does sub-region average pooling at 4 different scales. They do global average pooling over each feature map individually (1x1), they also divide the feature map into 2x2, 3x3 and 6x6 sub-regions and perform average pooling across each sub-region. Next they use a 1x1 convolution layer for each pyramid level for dimension reduction. This reduces the dimension of context representations to 1/N of the original one if the level size of pyramid is N. Finally at each pyramid level they upsample from these low-dimension feature maps to obtain the same dimensions as the original feature map (output of ResNet). PSPNet uses bilinear interpolation for this up-sampling. Next the upsampled pyramids are concatenated together, along with the original feature map. Finally one last convolution layer generates the final prediction map for the image.

Segmentation Results and Discussion

Figure 4: Brain Image Segmentation

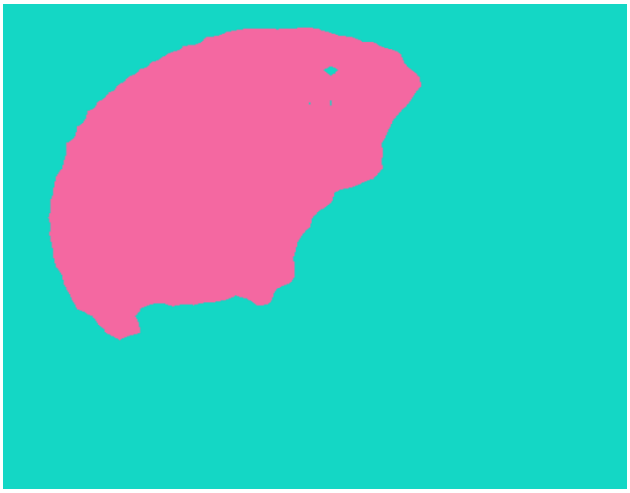

Figure 5: Beva Image Segmentation

Figure 6: Fibroid Image Segmentation

Figure 7: Breast Image Segmentation

Figure 8: Tumor Image Segmentation

Overall segmentation results across all 5 images from PSPNet were poor. At best PSPNet was able to segment the general shape of the body part from the background of the image (brain and breast images), however other images the general shape was lost completely and it was even difficult for myself to tell which image had just been segmented (fibroid and tumor image from Assignment 1). PSPNet was unable to segment any of the smaller regions within these body parts, even in the Beva image which had clear and distinct small regions of different intensity values. Even the segmenation on the relativly simple cropped tumor image that was provided in assignment 1 was very inaccurate, clearly indicating that this pretrained PSPNet is not suitable for accurate segmentation of medical images. A few other observations, in the brain image the brain shape is nearly perfectly segmented expect for a chunk in the bottom left hand corner, it is not clear why this section was excluded, that area appears to be very similar to the rest of the brain on the original image. The breast image segmentation is also very close to correct with respect to general shape, although again none of the finer details are recognized. The fibroid segmentation is by far the most noisy and least accurate, perhaps the lack of regions with uniform intensity caused this.

Thinking about how this model's architecture was designed and trained, it begins to make a little sense as to why the results are so poor. This PSPNet model was trained on the PASCAL VOC 2012 dataset. VOC 2012 is a dataset of 10,000 everyday images that have been segmented and each pixel has been labelled as one of the twenty recognized labels in this challenge. These labels have nothing to do with medical imaging, they are more focused on natural objects that you would see yourself; person, cat, bicycle, car, chair, sofa, etc. This brings up two significant issues with this pretrained PSPNet model for segmentation of these medical images. This model has been trained using a fixed set of labels, none of which apply to the five photos provided in this assignment, therefore when given these photos PSPNet is already looking for image features that are guaranteed to not be in the images, and is trying to decide whether to classifying each pixel as a person or a cat or something else far from the domain of internal body structure. The second issue is that the VOC 2012 images are RGB images, and therefore PSPNet was trained on three channel colour images, where as the medical images provided in the assignment are all greyscale. Therefore the ability that PSPNet has developed through training to use difference in colour (intensity in the three channels) is not very effective when working with greyscale images. Overall the issue is that the pretrained PSPNet was trained using colour images of natural everyday scenes with 20 fixed labels. Therefore PSPNet has learned what a human shape should be approximately, same with the shape of a bicycle, couch and cat, but it

has only learned these patterns for the 20 labels in the VOC dataset, none of which are found in medical images including the ones provided in the assignment. It is apparent that PSPNet has learned in general that regions of similar intensity tend to be considered one single object, this ability is demonstrated in the brain, beva and tumor segmentations. This serves as a good example of how a successful CNN based model designed for one specific task, will perform poorly if asked to work in another domain. The best possible way to get this pretrained PSPNet to produce successful segmentation on medical imaging would actually be to have it undergo a transfer learning procedure, where the PSPNet would be given a small training set of segmented medical imaging data to help refine its parameters to better work with greyscale medical images.

This argument was further supported when I tried an online segmentation demo based on [2], the demo can be found here `https://www.robots.ox.ac.uk/~szheng/crfasrnndemo/`. The model used in this demo is also trained on the PASCAL VOC 2012 dataset. The results it produces are as poor and sometimes worse in fact than the PSPNet (Figure 9)

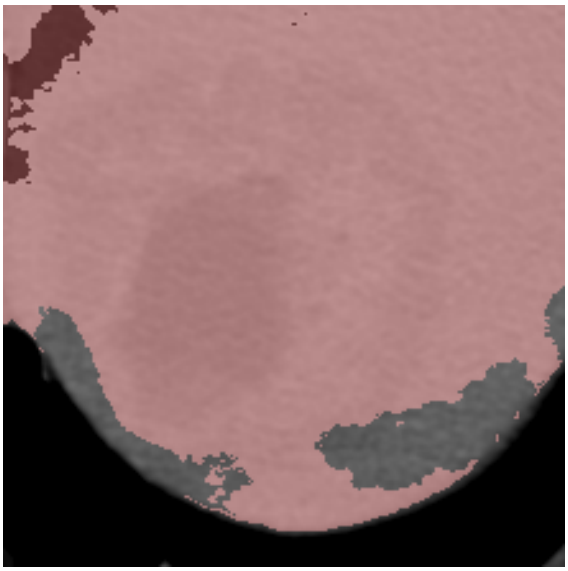

Figure 9: Tumor Image Segmentation from Demo of CRFs as RNNs

This further confirms the fact that PSPNet training on VOC 2012 is the main reason why it performs so poorly on the medical images. As a note, I suspect this analysis would have been nearly identical for the pretrained DeConvNet model, since DeConvNet was also trained on VOC 12 and would therefore have poor results like PSPNet and the online demo model [2] on these medical images.

U-Net Discussion
Although not explicitly confirmed I have provided an argument above that supports the notion that the pretrained DeConvNet would have performed very poorly in segmenting the 5 photos in this assignment. However I believe U-Net would have been much more successful for a number of reasons.

The first and most significant is that the pretrained U-Net model was trained on the cell segmentation dataset. This dataset comprises of greyscale images of cells. Immediately the fact that the images the U-Net model are trained on are greyscale indicates it could have an advantage over VOC 2012 trained models at accurately segmenting the images for this assignment. Further, cell shapes are more similar to the geometry of the distinct regions observed within the body parts in the assignment images, providing another advantage for the pretrained U-Net model over the VOC 2012 trained models, as an example, its unlikely learning how to segment the shape of a bicycle will help segment medical images.

The cell tracking segmentation data is formatted such that any pixel in an image is either deemed a cell or background, therefore the U-Net architecture is designed in a way that allows a user to change the number of classes (number of segments) predicted by the U-Net model. This is opposed to VOC 12 models that are strictly looking for any of the 20 VOC classes. This flexibility in the number of classes for a particular segmentation would have allowed for us (the user) to examine each of the 5 images, give a rough approximation as to how many segmentation classes we would want and then specify that in the U-net model, hopefully leading to more accurate results. Take the brain image for example, I would have said 1 class for the black background, 1 class for the white border, 1 class for the majority of brain tissue, 1 class for what I believe is a tumor on the left part of the brain and perhaps 1 extra class for extra border or fine detail pixels, for a total of 5 classes. I would be able to configure the U-Net to segment the brain image into 5 classes (unlike the 2 classes PSPNet segmented the image into).

Overall the training data and method for U-net is more similar to our data and segmentation goals. Therefore I think it is reasonable to think that the segmentations it would have produced would be far more accurate and detailed than any segmentation model trained on the PASCAL VOC 12 dataset.

Problem 4. Please see other submitted document

Problem 5. There are a few factors that could be the reason for the artificially high accuracy values. The first being how the datasets from different institutions were pooled together and split into the training, validation and test sets, this paired with the extreme model complexity of AlexNet and GoogLeNet (with respect to number of trainable parameters) allowed for the CNNs to learn about most or possibly all of the biases that would be present in the test set. Finally using imbalanced datasets (Belarus TB Public Health Program and Thomas Jefferson University Hospital) in the training set would have also contributed to this artificially high performance.

I believe the most significant reason is how the authors took images from 4 different institutions, pooled all the images and then divided them into training, validation and test sets. By splitting their data this way they removed the potential to test how well their model(s) generalize to images with different biases, then what is found in the training set. In the radiomics quality score, category 12 is all about validation, and the least significant form of validation is when you simply use data for validation that has the same theoretical underlying distribution as your training data (i.e. from the same institution(s)), which is the case in this paper. A more robust method for validation that would have more accurately assessed the model(s) accuracy would have been, for example, to train on the Shenzhen, China dataset, perform model selection with the Montgomery County, Md dataset, and validate model performance by combining the Belarus TB Public Health Program and Thomas Jefferson University Hospital datasets. By following this procedure the CNNs would not have been able to train and memorize all the biases and variations in images from the different institutions, since it would be trained on just images from one institution, and then subsequently tested on images from an institution it had no experience with. This method would have very likely lead to lower accuracy scores however they would have been more representative of the model(s) true predictive power.

This improper method for splitting the data into training, validation and test sets was easily exploited by the massive AlexNet and GoogLeNet architectures, particularly in the case of images from the two imbalanced datasets. AlexNet has roughly 60 million trainable parameters, GoogLeNet has roughly 6 million trainable parameters. For these experiments the training set was comprised of the images from only 685 patients (AlexNet and GoogLeNet were designed to train on ImageNet's 1.2 million photos), it's not unreasonable to think that AlexNet and GoogLeNet can simply memorize patterns from images in the training set, and not actually learn any underlying key representations that help with TB diagnosis. As an example imagine all of the present data in the study had roughly the same image format, same aspect ratio (before resizing), resolution and chest image positioning relative to the background, in other words from a distance all the images looked the same. This consistency would have allowed the CNNs to ignore where in the image the chest was positioned, and instead the CNNs would start to memorize what locations of the photos should be focused on without considering where in the chest those locations are (since the chest is always in exactly the same position in every photo). In this example, when the CNNs faced the test set, the test set images agreed with the image format of the training set and the CNN, ignoring where in the image the chest is exactly positioned would have been fine, and the AUC scores would be very high. Here we can see that the CNNs have been trained on where in the images are the regions that help distinguish and diagnosis TB, however a well trained model with good generalizability would instead be trained on where in the chest portion of the image are these descriptive regions regardless of chest region format relative to image. Suppose another hospital came forward after the test set scores were calculated, and had a new labelled dataset and wanted to see how well the CNNs worked, suppose these new images had a different aspect ratio, and therefore after resizing the chest portion of the image looked more compressed along the vertical axis then in the training set data. I believe that the CNNs from the paper would perform very poorly on this new dataset since it does not share all image properties to what was trained on.

The class imbalance of the Belarus TB Public Health Program and Thomas Jefferson University Hospital datasets can provide another example of how the data pooling was not a smart decision. Suppose that the Belarus TB Public Health Program images have a different aspect raio then the other 3 datasets and therefore when the images were resized to 256x256 image, chests from the Belarus dataset were more narrow

then any of the other dataset. It is possible that instead of trying to learn about underlying image features and representations that distinguish patients with TB from patients without, the CNN models simply learned that any image with narrow lungs should be classified positively, since the Belarus dataset was only positive instances. From this the models will all predict every instance from the Belarus dataset perfectly, even though they have not learned anything about how to distinguish TB from those data instances. Therefore the CNN models can exploit data imbalance by finding image biases unique to particular imbalanced dataset, memorize that bias, which could be unrelated to TB diagnosis and then simply check each image in the test set for that bias and immediately know its classification. For this reason, no imbalanced datasets should be used in the training set when using these massive CNN architectures, this is why the training, validation, test set scheme that I described in the first paragraph is favourable.

Overall, the root of the problem that lead to an inflated accuracy value was when the authors decided to pool data from the 4 institutions and then split that pooled data into training, validation and test sets. The better method would have been combine the 2 imbalanced datasets to create a 3rd balanced dataset and then keep those 3 datasets separate through model training, validation and testing. This procedure would have measured the general performance of these massive CNN architectures more accurately.

References

[1] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016.

[2] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. *CoRR*, abs/1502.03240, 2015.