# Extending Recurrent Conditional Random Fields and Convolutional Neural Networks to N-Dimensional Segmentation

**Abstract:**

*Conserving, capturing and reading text hidden away in fragile ancient works of literature has troubled historians for centuries. Most attempts to recover information from ancient documents have relied on physically altering the structure, often resulting in damage to the object itself. In the past year, ground breaking techniques have been discovered for capturing text hidden away in ancient documents all the while conserving the physical structure. Their methods involve capturing microComputed Tomography scans, constructing a volumetric model, and digitally uncovering the text. The software pipeline consists of multiple components – one of which is the process of isolating voxels located on scroll surfaces, more precisely referred to as segmentation. The aim of this paper is to apply machine learning to the segmentation process. This paper outlines a deep neural network where the efforts of Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Conditional Random Fields were combined to achieve state-of-the-art results for image segmentation.*

1. **Introduction:**

There is a number of ancient manuscripts around the world that remain unreadable. A majority of these manuscripts are too fragile to open, and the literature remains undiscovered. While there are many types of ancient manuscripts, this paper focuses on scrolls – in particular, those scrolls that are found in the ancient city of Herculaneum. After the eruption of Mt. Vesuvius, a library containing hundreds of scrolls from antiquity was buried under volcanic ash.

Scrolls exist in three-dimensional space, meaning that, in order to unwrap the scroll, a volumetric model must be generated. After the model is generated, the voxels must be pruned in a way so that only the voxels that contain relevant information remain. Segmentation is the process of isolating only those voxels located on papyrus surfaces where actual text is found.

The software process developed in Seales et al. includes a current segmentation algorithm contained within the in-house developed *Volume Cartography* application. The application displays a GUI where the user can draw seed points on a local scroll layer, where afterwards a spline function is interpolated between seed points. The application then makes use of an algorithm that attempts to maintain the location of the spline function on the scroll layer through the z-dimension. The algorithm constructs eigenvectors around local sub-volumes and makes a greedy decision as to where the next coordinate (in the z-dimension) located on the scroll surface is. While the application produces results, it is time consuming because it requires humans to sort through every slice and verify correctness. The segmentation algorithm is also unable to handle odd scenarios that arise in the volumetric scan.

****

Figure : An example crop from a single scan slice. The fourth layer from the left shows an example of torn papyrus

The scrolls found at Herculean were written on papyrus, and papyrus raises two essential problems during the segmentation process. Papyrus is constructed by pressing together crosshatched pieces from the papyrus plant. Due to the nature of the *pressing* and the general strength of papyrus, the scrolls are easily torn and/or shredded. See Figure 1 for an example of shredded papyrus. Torn surfaces make the segmentation algorithms greedy decision more ambiguous, and thus results are less consistent. Furthermore, the sheets of papyrus lack rigidity and are easily folded. A fold in the *x,y* plane doesn’t cause any issues for the segmentation algorithm. It is when the papyrus folds inwards to the *z-dimension* that issues arise. The segmentation application moves perpendicular to the *z-dimension* and only generates single points for each *z-slice.* When a papyrus surface folds inwards in the *z-dimension* the segmentation algorithm tends to lose track of the surface and/or doesn't generate multiple spline functions within a single slice.

By applying machine learning to the segmentation process, the issues raised above can be resolved. Machine learning, and Convolutional Neural Networks (CNNs) in particular, have shown state-of-the-art performance in image classification, object recognition, and pixel-wise classification – even out performing human standards in some cases. For this paper, I apply CNN's for voxel-wise classification on volumetric data – otherwise known as volumetric image segmentation. The application of deep networks towards segmentation on volumetric data remains a relatively undiscovered field. Through the use of three-dimensional receptive fields, CNNs can offer a level of perception not viewable by humans. Since a three-dimensional CNN can offer new fields of view, the problem of papyrus shredding and folding can be resolved. A level of surface cohesiveness through the *z-dimension* can be viewed by a three-dimensional receptive field. For example, the crosshatching patterns are not viewable through a single *x, y* view, but given a view of all three dimensions, the CNN can perceive a continuous scroll surface.

1. **Related Work:**

There is a collection of notable published papers that applied CNN for image segmentation – some of which are two-dimensional networks and more recently, others extended the networks to three-dimensions. SegNet makes use of an encoder-decoder architecture where shallower downsampling layers are forward passed to deeper upsampling layers [4]. Badrinarayanan et al. claimed that through the use of the decoder stack, predictions appear less blocky. Long et al. showed how fully connected convolutional layers coupled with upsampling can yield positive results for two-dimensional image segmentation. More recently Milletari et al. applied three-dimension CNNs for volumetric medical image segmentation. The CNN was composed of convolutional layers wherein the receptive fields consisted of three dimensions – thus capturing information in all three dimensions of volumetric scans. A similar implementation was developed by Ji et al. where a three-dimensional CNN classified human actions. Similar to Karpathy et al. the data consisted of video frames, only instead of two-dimensional input frames, the frames were stacked into three-dimensional volumes.

This paper attempts to apply parts of the network developed by Zheng et al. extended to three-dimensions. The recurrent neural network conditional random field [2] is an extension of the work done by Chen et al. where a fully connected Conditional Random Field (CRF) served as a post processing step after a CNN for semantic image segmentation. CNNs outputs are not sufficiently localized for accurate segmentation, but CRFs can refine local information [2]. Zheng et al. refined this technique by constructing an end-to-end trainable network that makes use of a CNN connected to a recurrent CRF – referred to as a RNN-CRF. Since the network is constructed in an end-to-end fashion, weight optimization through backpropagation can be conducted in a single system.

1. **Model:**

**3.a CNN:**

To segment a volume, the output volume must have the same dimensions as the input volume. CNNs often make use of max pooling layers or downsampling layers through non-unit strides. Downsampling is the process of reducing image dimensions to capture local information. Since the dimensions must be maintained in the prediction volume, downsampling is an issue for volumetric image segmentation. Not making use of downsampling is also an issue because the strength of capturing local information is not put to use. This network makes use of downsampling in order to perceive local information, followed up by upsampling in order to perceive global information. Upsampling layers are commonly incorrectly labeled as deconvolutional layers, but the correct term for upsampling layers is convolutional transpose layers. Convolutional transpose layers make use of padding the input volume so that the output volume can have larger dimensions – see Figure 2 as an example convolutional transpose layer. The general structure of the CNN in this paper is composed of three convolutional layers followed by three convolutional transpose layers. See Figure 3 for a diagram of the CNN. Each convolutional layer makes use of a stride equal to two, thus halving the volume dimensions. Each convolutional transpose layer also makes use of a stride equal to two so that the dimensions of the final output volume are equal to the input volume.

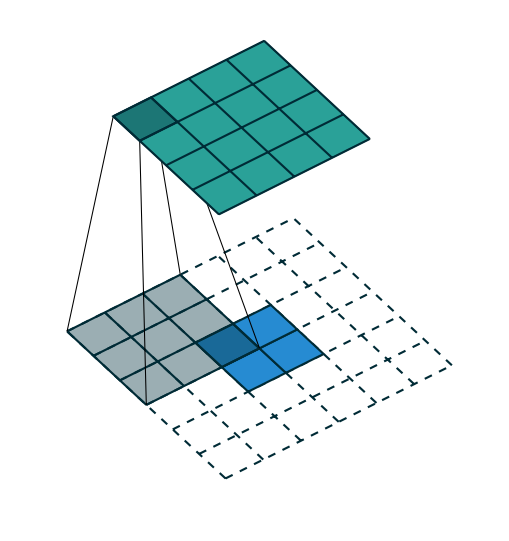


Figure : Convolutional transpose layers make use of padding to output volumes with larger dimensions [8].

The CNN also makes use of a handful of other modern techniques. In order to prevent learning divergence, convolutional filter weights are initialized using Xavier Initialization. Xavier Initialization initializes weights depending on the number of incoming and outgoing connections. If weights are initialized too small, then there is insufficient signal strength through the network. If weights are initialized too large, then the network predictions diverge into random noise. Xavier Initializations resolve both of these problems.

After several experiments, I found that defining receptive field sizes to roughly one fourth of the input volume dimensions produced the best results. Convolutional layers share weights across the entire input volume. For example, the filter for the first convolutional layer has the dimensions: [z, x, y, number of input channels, number of output channels] → [14, 14, 14, 1, 16]. The number of input and output channels is more correctly labeled as the size of the feature map. Each individual channel uses a single convolutional filter across the entire volume – this can lead to blocky predictions (discussed further in section 4).

All layers make use of batch normalization. Batch normalization is placed after each linear Rectified (ReLu) activation function in order to prevent divergence. Dropout is applied after the final layer to prevent overfitting. Dropout is a mechanism where a random number of connections (specified by a probability) in the final layer are disconnected. The probability assignment for dropout is a tunable hyperparameter.

****

Figure 3: The CNN consist of three convolutional layers followed by three convolutional transpose layers.

**3.b Conditional Random Fields:**

A Conditional Random Field (CRF) is a statistical modeling method than can be applied to machine learning for structured predictions. In the case of this model, the CRF is responsible for voxel-wise classification. In a fully connected CRF, the energy function of a label assignment x is given by [2]:

The unary term measures the cost of voxel *i* taking the correct label. The pairwise term measures the cost of assigning labels to voxels *i* and *j* simultaneously. The unary term is handled by a convolutional filter. The pairwise term is handled by implementing a Gaussian filter, or more precisely, a bilateral filter, where the formula for the labels assigned to voxels *i* and *j* is given by [2]:

**3.c** **Recurrent Neural Network – Conditional Random Field:**

The Recurrent Neural Network Conditional Random Field (RNN-CRF) is nearly identical to the model developed by Zheng et al. extended to three dimensions. The general architecture of the RNN-CRF inputs three volumes: the original volume, the output volume from the CNN (not normalized), and the normalized output volume – initially from the CNN, but after the first iteration, the normalized output of the RNN-CRF is fed back into the RNN-CRF. The softmax function is used as the normalization function for this network. See Figure 5for a diagram of the general network architecture.

Zheng et al. developed a clever implementation of a CRF by making use of convolutional layers, labeling the CRF implementation as *a mean-field iteration as a stack of convolutional layers* (see Figure 4). There are five steps to the CRF: message passing, re-weighting, compatibility transform, unary addition, normalization. After the final step, the output from the normalization is fed back into the message passing step – this is the *recurrent* component of the RNN-CRF (see Figure 5). The number of iterations is a tunable hyperparameter.

**3.d** **Message Passing:**

The message passing step of the CRF is where bilateral filters are applied to the input volume. bilateral filters rely on capturing two essential components from the input volume: voxel intensity values and voxel locations. The responsibility of the bilateral filter is to penalize voxels that have similar intensity values but are located across segmentation lines. The network developed in this paper implements message passing in two convolutional layers. Both layers are fully connected, so in both layers the size of the receptive field is equal to the volume dimensions. The first layer is a fully connected convolutional layer with learnable filters – this learns the correct voxel intensity values. The second layer is a fully connected convolutional layer with constant filter values. The weights within this filter are equal to the distance between each voxel – this measures pair-wise voxel distance values. The input volume has a single input channel and *M* output channels.

**3.e** **Re-weighting Filter Outputs:**

The purpose of the re-weighting step is to take the weighted sum of each filter weight from the previous *M* feature maps. This step can be implemented by simply utilizing a single convolutional layer where the filter size is equal to [1, 1, 1, *M*, *l*]. For this step there are *M* input channels from the message passing step and a single output channel.

**3.f** **Compatibility Transform:**

The compatibility transform step is responsible for assigning penalties to voxels that share similar properties but are assigned different labels. This is defined in the Potts model as *u(l,l') = [l != l']* [2]. However, this model is not flexible. Given a penalty in the Potts model, the penalty is always the same constant value. The compatibility transform provides flexibility through utilizing a convolutional layer with a receptive field size of [1, 1, 1, *l*, *l*]. Therefore, the weights will learn to apply different levels of penalty depending upon voxel intensity. This step should have *l* input channels and *l* output channels.

**3.g** **Unary Addition and Normalization:**

In the unary addition step, the output from the compatibility transform is subtracted element-wise from the non-normalized output from the CNN. Subsequently, the output volume is normalized via the softmax function. After normalization, if the iteration number is less than the specified hyperparameter, then the normalized output is fed back into the message passing step.

****

Figure 4: The mean-field iteration is a stack of convolutional layers. Input C is the output from the CNN, input N is the normalized output, and input I is the original image.

****

Figure 5: The RNN-CRF includes a recurrent feedback loop. Input I is the original image, and input C is the output from the CNN.

1. **Data:**

**4.a** **Input Data:**

The input data for this experiment was collected from a sample scroll – referred to as a Phantom scroll because it is not an actual scroll from Herculaneum. The Phantom scroll is a scroll constructed by humans for testing purposes. The dimensions for the entire scan is [3820, 3820, 3398]. The total size, after compression, on this dataset is toughly 68GB. Due to hardware limitations, training on the entire scroll is out of question. Therefore, the scroll was cropped into a more manageable size. The training session had an entire training volume of dimensions [396, 396, 100] and an input sub-volume of dimensions [54, 54, 54]. The microComputed Tomography scan was set to capture one voxel for every 14 micrometers. Each volume crop was sliced in half along the z-dimension so that one half was used as training data and the other half used as prediction data.

**3.b** **Ground Truth**

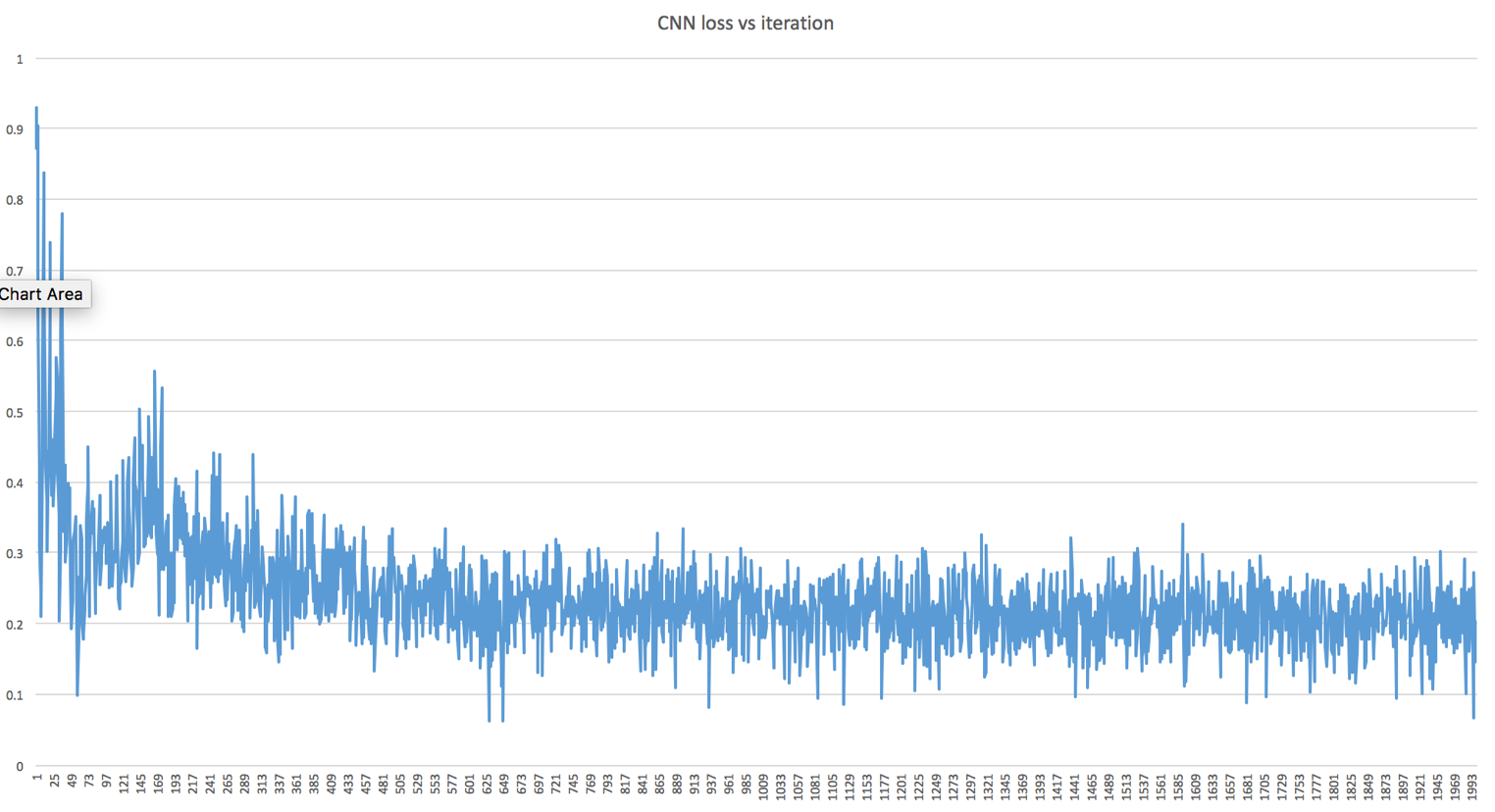
Ground truth for the cropped volumes was collected by using the already developed segmentation method (outline in section 1) [1]. To recap, the *Volume Cartographer* application requires the user to define a set of seed points. Subsequently a spline function is interpolated between the seed points. Once segmentation begins, local sub-volumes are passed into an algorithm which makes a greedy decision on the location of the next surface voxel. The segmented points generated by *Volume Cartographer* are placed in PLY files. The points are sparse, and therefore must be filled in. A data script executes a recursive algorithm that first fills in lines between points – making each surface a continuous line – and then expands the line to approximately the width of the actual scroll surface. Expanding each line generates a probability gradient where the highest probability is located at the center of the layer, and the lowest probability is located at the edge of the layer. One problem with this network is that fixed width expansion results in inaccurate ground truth since actual scroll layers are variable width.

1. **Experiments:**

The entire network and all data handling was written using Google's Tensorflow deep learning API. The learning rate was set to 0.001. The number of training iterations was set to 2,000. The batch size was set to 3. Keeping consistent with the modern choice for image segmentation, the softmax cross entropy loss function was used after the final layer. The Adam Optimizer was used for optimizing weights through gradient descent.

Due to the nature of the convolutional filter, predictions can appear blocky. A *smoothing* method was developed to overcome blocky predictions. The prediction sub-volume is half of the total volume of the input sub-volume so that edge voxels share the same amount of information as center voxels. Input volumes overlap with a step of nine, and therefore predictions are also overlapped. Overlapping predictions improves the smoothness of prediction volumes – every voxel is an average of nine overlapping volumes.

There were two training sessions. The first network was composed of the network above – CNN-RNN-CRF – and another network that was strictly a CNN. Training these two networks offered insight to the level of improvement the RNN-CRF offered. Training was conducted on a NVIDIA GeForce GTX 1070. Training the CNN-RNN-CRF took roughly one week. Training the CNN took roughly 4 hours. After conducting 2000 iterations, the cross entropy loss function over time computed nearly identical values. The graphs of loss over iteration can be found in FIGURES XY below. The minimum loss value for the CNN was 6.18E-02, and the minimum loss value for the CNN-RNN-CRF was 7.85E-02. Based on the loss function, both networks were able to learn to the same extent, but predictions appear different. Example two-dimensional slices can be found in FIGURES XYZ below. The CNN-RNN-CRF prediction slice appears to better match the width of the scroll layers. This is most likely attributed to the Gaussian filter included within the CRF. The CNN-RNN-CRF was also able to better predict the shredded surface (shown in figure 1 and FIGURE X below).



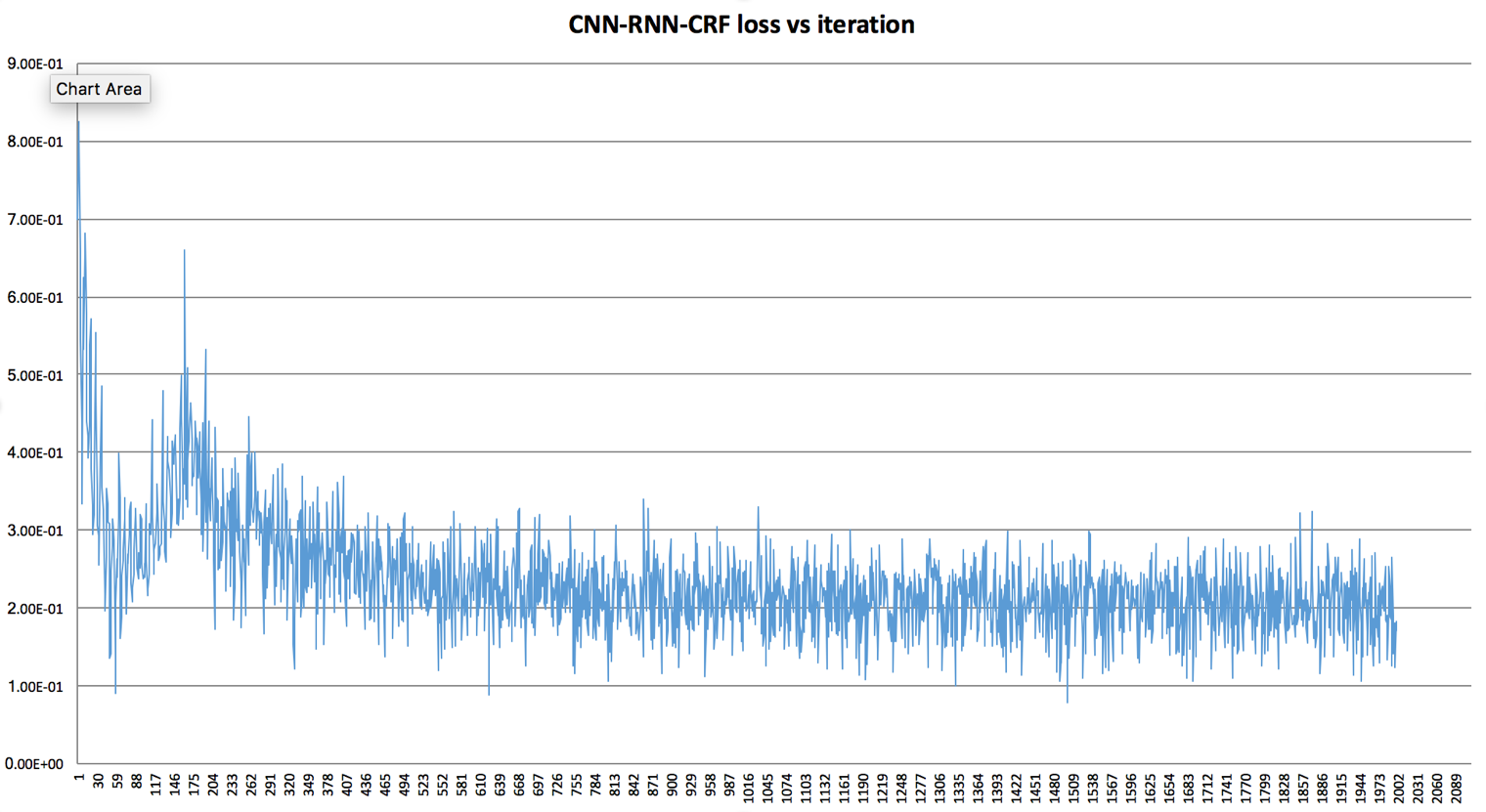




Figure 3: the actual input slice.



Figure 4: prediction slice from the CNN.



Figure 5: prediction slice from the CNN-RNN-CRF.

1. **Conclusion:**

Seales et al. published ground-breaking results for the discovery and preservation of ancient scrolls. One component of the scroll unwrapping pipeline is the process of segmenting voxels located on scroll surfaces. The current software pipeline can handle segmentation, but is time consuming because it requires a human to sort through every slice and verify its correctness. In this paper, we have discovered the potential in applying machine learning to automate the segmentation process. One drawback in the current deep learning segmentation implementation is the fixed width ground truth. The actual scroll layers vary in width. Also, since *Volume Cartographer* generates points that are not always placed in the center of the scroll layer, expanding the width of the ground truth can often expand past the true scroll surface. The RNN-CRF has shown promising results in picking up variable width layers, but there is still work to do to improve ground truth data. Future endeavors include applying a deep network to the current unwrapping pipeline, and/or even developing an entirely new and fully automated segmentation process through machine learning.

**References:**

1. Seales, William Brent, Clifford Seth Parker, Michael Segal, Emanuel Tov, Pnina Shor, and Yosef Porath. "From damage to discovery via virtual unwrapping: Reading the scroll from En-Gedi." *Science Advances* 2, no. 9 (2016): e1601247.
2. Zheng, Shuai, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. "Conditional random fields as recurrent neural networks." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529-1537. 2015.

|  |
| --- |
| 1. Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. "Semantic image segmentation with deep convolutional nets and fully connected crfs." *arXiv preprint arXiv:1412.7062* (2014). |

1. Badrinarayanan, Vijay, Ankur Handa, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling." *arXiv preprint arXiv:1505.07293* (2015).
2. Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431-3440. 2015.
3. Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation."*arXiv preprint arXiv:1606.04797* (2016).
4. Ji, Shuiwang, Wei Xu, Ming Yang, and Kai Yu. "3D convolutional neural networks for human action recognition." *IEEE transactions on pattern analysis and machine intelligence* 35, no. 1 (2013): 221-231.
5. Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).
6. Karpathy, Andrej, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. "Large-scale video classification with convolutional neural networks." In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725-1732. 2014.