U-Net Compression Using Knowledge Distillation

Karttikeya Mangalam and Dr. Mathieu Salzmann

Master's Semester Project, EPFL, Switzerland
karttikeyamangalam@gmail.com,
https://karttikeya.github.io/

Keywords: Model Compression, Knowledge Distillation, Dark Knowledge, U-Net, Biomedical Segmentation

1 Introduction

1.1 Motivation

Recently, deep learning has achieved path breaking results on several computer vision tasks. Neural networks have replaced many older 'traditional' computer vision techniques and have ushered in the era of large scale data driven learning. State of the art models for many common vision tasks are fine-tuned and highly optimized neural networks such as Image Recognition [3] (VGG16, VGG19), Human Pose estimation [4] (Stacked Hourglass Networks) and Segmentation [15] (U-net). However, the increasing depth of such models also results in a higher storage and runtime complexity, which restricts the deployability of such very deep models on mobile and portable devices, which have limited storage and battery capacity. Moreover, the large amount of hyper parameters also prohibit proper understanding the learned representations.
Hence, exploring techniques that allow us to train a smaller model that is similar in architecture to the original model but has significantly smaller number of parameters would allow us to overcome the above challenges. Additionally, for such an attempt to be truly successful in reducing the network size (in terms of the number of parameters) it should also achieve performance similar to the original heavier model on benchmark datasets with minimal training time overhead.

1.2 Related work

Most neural network compression approaches fall in three broad categories: weight quantization, architecture pruning and knowledge distillation. The first approach attempts to compress by minimizing the space footprint of the network by utilizing less space for storing the value of each parameter through value quantization. Even with just as small of two-three bits per parameter, these methods can approach state of the art accuracies [8][9].
Le Cun et al. [10] pioneer the approach of architecture pruning whereby they

train a full network and then they do away with the neurons with near to zero activations. Later on, Han et al. [11] propose a method to jointly learn weights and network connections and demonstrate an architecture pruning approach without performance loss. The work presented in this report falls in the third category of knowledge distillation. Approaches in this category typically train a heavier network to start with and then use this 'teacher' network to train a smaller 'student' network through knowledge transfer. We discuss there approaches in detail in the next subsection.

Knowledge Distillation First attempts in the direction of knowledge distillation were made by Caruna et al. [6]. They present the algorithm MUNGE for model compression and investigate the model complexity- RMSE error for MLP. Later on, Hinton et al. [7] propose a more general technique for distilling the knowledge of a network utilizing the predicted probability distribution of the teacher model to train the smaller (student) model. This forms the basis of our exploration and we will revisit this later. Other approaches such as Deep Model Compression by Bushan et al. employ a noise based regularizer dependent on the predictions of the larger model while training the student model. Bengio et al. [12] propose an alternative knowledge distillation training approach that utilizes intermediate deep representations learned by the teacher model in addition to the final predicted probability distributions to train the student network. Also, several application focused methods for knowledge distillation have been developed such as compression for face model identification [2], for object detection [1] and face verification [13]. Recently, Lopes et al. propose a 'data-free' knowledge distillation procedure that utilizes on a small meta-database alongwith the trained teacher model to train the smaller model agnostic to the large database originally used to train the teacher model.

All the above three proposed approaches for model compression are largely orthogonal to each other and can be employed simultaneously to achieve $\tilde{5}$0x model compression as demonstrated by Han et al. [14].

2 Preliminaries

In this section, we describe some of the architecture, conventions and datasets upon which our work heavily relies upon.

2.1 The U-net Architecture

The original U-net architecture as proposed by Ronneberger et al. [15] is a fully convolutional network with skip connections that comprises of a contracting path and an expansive path. Kindly refer to Figure 1 for details. The original u-net architecture comprises of channel depth of 64 at the first level that doubles each time for 4 consecutive stages reaching 1024 at the bottom level and reduces back to 64 in the top stage of the expansive part. Later when we introduce the batch

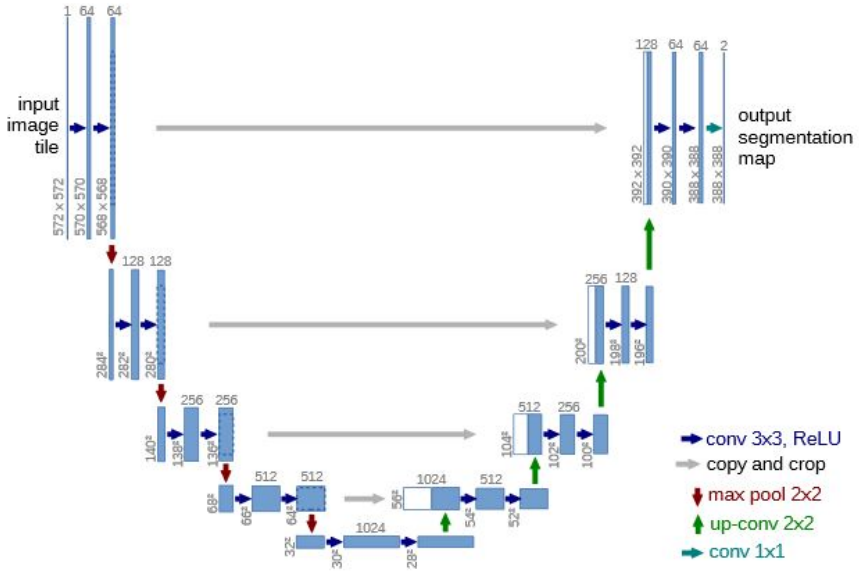

Fig. 1. Illustration of the U-net architecture. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operation [15].

normalization layer as described in Section 3.9 in the expansive part before the ReLu activation. So, the operation denoted by the blue arrow in the above figure becomes (conv 3x3, BN Layer, ReLU) instead of (conv 3x3, ReLU).

2.2 Dataset Used

Since, U-net is widely used for segmentation in biomedical images, we primarily use the Mitochondria Segmentation dataset in Electron Microscopy stacks [16]. It represents a 5x5x5 micrometer section taken from the CA1 hippocampus region of the brain, corresponding to a 1065x2048x1536 volume. The resolution of each voxel is approximately 5x5x5nm. It consists of two separate sub-volumes for training and testing. Each sub-volume consists of the first 165 slices of the 1065x2048x1536 image stack. A sample image and it's corresponding annotation is shown in Figure 2. We pose the problem in the framework of binary segmentation where white represents a pixel that is part of a cell's mitochondria and black represents all other pixels. Note that different images represent the same 3D structure of the brain viewed through different cross-sections. Alternatively, to verify some of our results on another dataset, we have employed the Synapse segmentation dataset in Electron Microscopy stacks [17].
Naturally, the problem facilitates a 3D deep learning approach but for the pur-

poses of this work, we take a 2D approach, processing the slices(images) one by one. The reason for the above design choice is that using 3D deep learning would require require a major overhaul of the original U-net architecture thereby limiting the use of such an effort to only 3D problems. In addition to this, even with the 2D constraint several interesting observations are discovered which could now be utilized for developing the 3D U-net in future.

2.3 Experimental Setup

Consider the training examples to be denoted by (x_i^t, y_i^t) and the network's estimate of the output to be $\hat{y_i^t}$. Similarly, the testing examples to be denoted as (x_i, y_i) and the network's estimate of the output to be $\hat{y_i}$. Also, consider the cross entropy loss function $\mathcal{H}$: $(\mathbb{R}^{N\times N}, \mathbb{R}^{N\times N}) \to \mathbb{R}$ to be defined as,

$$\mathcal{H}(\mathbf{x}, \hat{\mathbf{x}}) = -\sum_{(i,j)} \mathbf{x}_{ij} \log(\hat{\mathbf{x}}_{ij}) + (1 - \mathbf{x}_{ij}) \log(1 - \hat{\mathbf{x}}_{ij})$$

Moreover, suppose that the training losses are averaged over α_t individual consecutive examples. Similarly, assume that the testing losses are averaged over the entire testing dataset every n_t training iterations and then β_t of such averages are averaged. Also, assume that the testing set consists of n_{test} number of examples. The reported training loss $\mathcal{L}^*_{train}$ is then described as follows:

$$\mathcal{L}^*_{train} = \min_k \frac{1}{\alpha_t} \sum_{i=k}^{k+\alpha_t} \mathcal{H}(y_i^t, \hat{y^t}_i)$$

Let us denoted the testing loss at the training iteration i as $\mathcal{L}^i_{test}$ defined as,

$$\mathcal{L}^i_{test} = \frac{1}{n_{test}} \sum_{j=1}^{n_{test}} \mathcal{H}(y_j, \hat{y}_j)$$

where, $\hat{y}$ are output of network's final softmax layer after being trained for i iterations. The reported testing loss $\mathcal{L}^*_{test}$ is then described as,

$$\mathcal{L}^*_{test} = \min_k \frac{1}{\beta_t} \sum_{j=k}^{k+\beta_t} \mathcal{L}^{jn_t}_{test}$$

The values of the parameters α_t,β_t and n_t are specified for each experiment and are chosen so as to allow standardization across the work and meaningful insights wherever possible. All the experiments are performed on a single Titan X GPU with PyTorch as the DL framework and CUDA 8.0 backend.

3 Distilling the U-net

In this section, we describe our several attempts at distilling the U-net architecture, including both the failures and the successes. We draw fruitful insights from our failures by promptly investigating and reporting the probable reasons along with explaining the intuition behind the successful approaches.

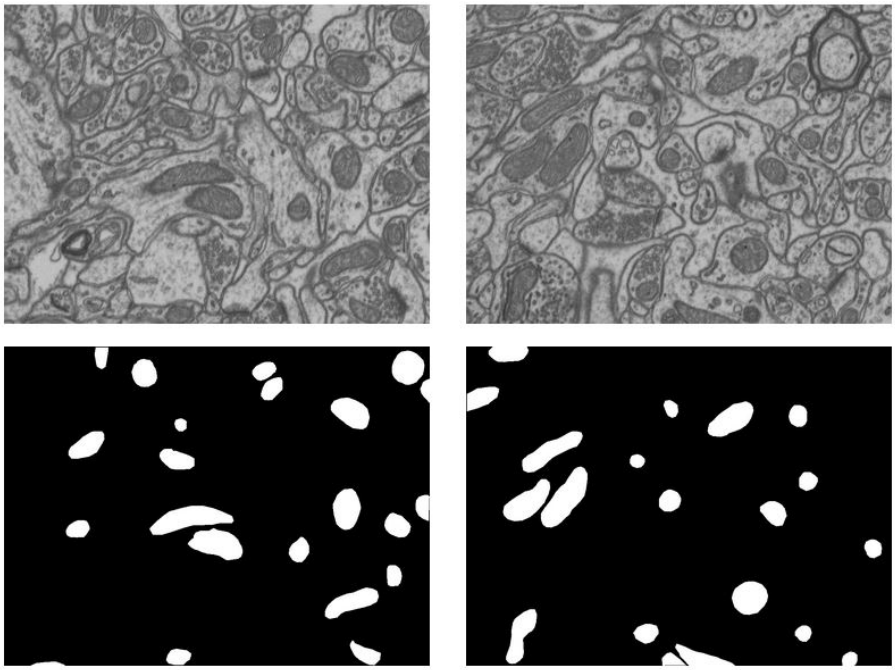

Fig. 2. A slice and it's corresponding annotation from the Mitochondria Segmentation in Electron Microscopy Dataset [16].

3.1 The Student Architecture

To come up with a suitable student architecture to use for in knowledge distillation, we reduce the channel depth at the first stage of the U-net architecture keeping the doubling pattern to be the same . In particular, we start with the depth of 64 as in the original u-net and keep halving it each time until the perform worsens significantly. For ease of notation, we call a U-net architecture with a channel depth of k in the first layer as the k-Unet. Referring to Table 1, we see that the U-net works surprisingly well upto just 7% of it's original parameter (4-Unet). We also reconfirmed this result on the Synapse detection dataset as described in Section 2.2. So, throughout this work, we use the 2-Unet/1-Unet as our student model and the 4-Unet as the teacher model for knowledge distillation. Observe that, this simple investigation in itself provides over **$\tilde{1}$3x compression** for the U-net.

Note that, the reason we tweak the channel depth keeping other model parameters such as the kernel size and the number of stage the same is two-fold. Firstly, reducing the number of stages in the architecture also reduces the reception field of the CNNs, the loss of which cannot be gained from the additional transfer knowledge from the teacher model. Secondly, the original architecture uses kernel sizes of 2x2 & 3x3 and reducing them further is not possible. Hence we have

Starting Channel Depth	Test Loss	Train Loss	#Iterations
128	0.1170	0.0281	63,000
64	0.1021	0.0256	80,000
32	0.0871	0.0220	125,000
16	0.0822	0.0220	150,000
8	0.0830	0.0221	280,000
4	0.0974	0.0286	300,000
2	0.8227	0.3423	290,000
1	0.8337	0.3438	320,000

Table 1. Performance of different U-net architectures on varying the channel depth in the first layer. The loss parameters used are $\alpha_t = 5000$, $\beta_t = 25$ and $n_t = 500$. The reported number of iterations is where the minimum for the testing loss is achieved.

worked on reducing the channel depth, which is inversely proportional to the total number of learnable parameters, keeping other model parameters constant.

3.2 Vanilla Training of U-net: Memory issues

Data Augmentation Since, the training data is small relative to the model complexity, it is supplemented through flipping the original image both horizontally and vertically and also through elastic stretching. Since, elastic stretching of an image is a stochastic process, it inherently creates a very large number of unique images (practically unlimited) that have minor variations in between them and can them be used to train the Unet.
A problem that the above augmentation procedure presents is that of storing the teacher network's predicted probability distribution $\mathcal{P}_t^*$. Since on each run, the U-net uses a different set of images to train owing to the stochasticity of elastic stretching process, $\mathcal{P}_t^*$ cannot be stored beforehand to be used as 'soft' labels for the student network but need to be generated *in-situ* during the training process. This requires loading the teacher U-net in the GPU memory while training the student U-net and can present a memory issue with large models.

Overlap tile strategy U-net model is a fully convolutional network and thus the memory footprint depends directly on the size of the image used. As discussed in [15], during the training phase we process the image in blocks (patches of the original image) that are augmented as described above and are treated as separate image for all practical purposes even though they form parts of a larger slice. During the prediction phase, these blocks are passed independently through the network and then their corresponding predictions are stitched together to form the complete output image. Refer to Figure 3 for an illustration of this technique.

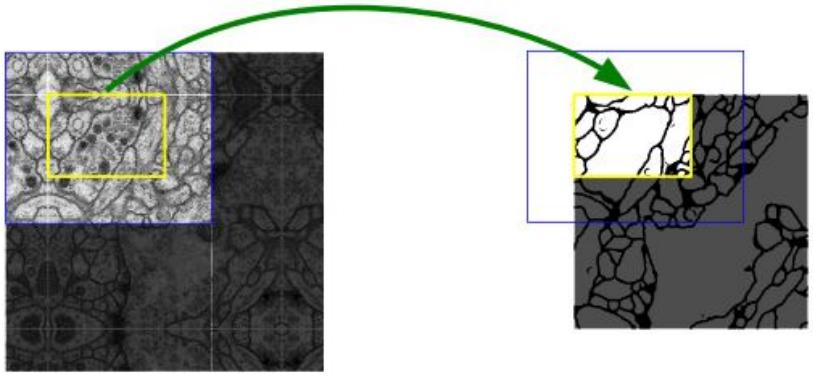

Fig. 3. Illustration of the overlap tile strategy for saving memory on large images. The blue area is the input for prediction in the yellow area. Mirroring is used for interpolation in missing data. [15]

3.3 SoftMax temperature

Since we are employing the Unet as a classifier, the final layer is the "softmax" layer that is typically used for classification in neural networks. It computes the probability q_i for each class, given it's corresponding logit z_i, by comparing z_i with other logits as follows [7],

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

Usually, the softmax temperature T is set to 1 both during the training and the testing phases for an architecture. However, if need be, this temperature can be changed to a higher value than 1 to produce a 'softer' probability distribution. Also, as noted in [7], using a value other than 1 also scales the gradients by $\frac{1}{T^2}$ and so the corresponding soft loss needs to be multiplied by T^2 when being added to the hard loss for guided training as described in Section 3.7.

3.4 Distillation with soft targets

We follow the procedure described in [7] for distillation using the predicted probability distribution of the teacher network. First, the teacher network, 4-Unet is trained from the raw mitochondria dataset with the softmax temperature set to 1. Further, teacher network's predicted probability distribution, $\mathcal{P}_t^*$, generated at a softmax temperature $T_{transfer} > 1$ is used to train the student network. We use the cross entropy loss between $\mathcal{P}_t^*$ and that probability distribution predicted by the student network,$\mathcal{P}^*$ (See 2.3 for details) as the teaching signal. The student network is the 2-Unet being trained from scratch with $\mathcal{P}_t^*$ as soft labels

$T_{transfer}$	0.1	0.25	0.5	1	2	4	5	7	10	15	20
Test loss	Div.	Div.	Div.	0.111	0.480	0.104	0.610	0.481	0.480	0.480	0.134
#Iterations	-	-	-	15,000	15,000	50,000	50,000	9000	14,000	14,500	48,800

Table 2. Cross entropy test loss for soft training of 2-Unet. The teacher model is 4-Unet trained for 300,000 iterations with 0.0910 as the test loss. The above reported losses are the minimum across the test losses with no averaging. Div. indicates that the training diverged on several hyper parameter combinations.

with the softmax temperature $T_{transfer}$ same as that of the teacher model. After finishing the training for 2-Unet, it's temperature is set back to 1 for testing and prediction. Refer to Table 2 for details.

3.5 Stochasticity in Training the 2-Unet

The above results indicate extremely low correlation between network's performance and softmax temperature used in the distillation process which is odd and warranted further investigation. The results of re-runs of the above experiment are reported in Table 3. As the results indicate, the distillation procedure results seems to be stochastic and the results are non-repeatable. For example, none of the results remain constant throughout the three trials except for temperature as 2, when the 2-Unet gets stuck at the 0.48 loss each time. This led to the suspicion that the cases where 2-Unet actually trains are not due to the distillation procedure but because of the random good initialization of the model. To further confirm our suspicion, we re-train the 2-Unet multiple times without supervision of the teacher model directly from the Mitochondria dataset. Referring to Table 4, observe that three out of the ten times, the 2-Unet trains upto the best possible performance while the rest seven times it completely fails to train and does no better than a randomly initialized model. Thus, we conclude that training the 2-Unet is an inherently stochastic process and the final performance depends heavily on the initialization weights of the model. So, this conclusion also validates our suspicion that the seemingly bizzare pattern observed in Table 2 is nothing but an artifact of this stochasticity and has nothing to do with the softmax temperature itself.

Physical significance of 0.48 loss As can be observed in Table 3, whenever the distillation procedure fails, it gets stuck the 0.480 test loss. Physically, this means that the network's prediction for each pixel to belong to class i is exactly the percentage of class i in the overall dataset and is actually independent of the pixel's value and it's surroundings. In particular, in our case, the predicted probability distribution $\mathcal{P}^*$ takes the form,

$$\mathcal{P}^*(\mathbf{x}_{ij}) = \begin{cases} 0.947 & \mathbf{x}_{ij} \in \textit{background} \\ 0.053 & \mathbf{x}_{ij} \in \textit{foreground} \end{cases}$$

$T_{transfer}$	Testing Loss (avg)			Testing loss (min)		
	Trial 1	Trial 2	Trial 3	Trial 1	Trial 2	Trial 3
1	0.481	**0.212**	0.483	0.480	**0.210**	0.480
2	0.488	0.488	0.485	0.480	0.480	0.480
3	0.482	0.482	**0.223**	0.480	0.480	**0.215**
5	**0.123**	0.482	0.489	**0.097**	0.480	0.480
6	**0.151**	0.525	0.482	**0.126**	0.496	0.480
10	**0.120**	0.482	0.482	**0.093**	0.480	0.480
15	**0.209**	0.482	0.491	**0.201**	0.480	0.480

Table 3. Three trials of distillation with 2-Unet as the student model. The parameters for average test loss are $\beta_t = 25$ and $n_t = 500$. The bold values indicate breaking the 0.48 test loss barrier.

for the Mitochondria segmentation dataset in Electron Microscopy stacks (refer to section 2.2).

An unexpected benefit While the distillation procedure seems to provide no more chances of convergence than pure random chance, when the training does converge, it converges much faster in teacher guided training than the vanilla training process. This effect can be seen on comparing iteration values between Table 2 and Table 1.. Around **20x speedup** is achieved in convergence time by using the teacher model for guided training (15,000 iterations in teacher guided training vs. 300,000 iteration in vanilla training).

3.6 Soft-training the 1-Unet

To avoid the problem of re-running the experiments to deal with random initialization artifacts of the 2-Unet we also experimented with the 1-Unet on similar lines as described in Section 3.4. On higher temperatures (above 20) the 1-Unet, with less than 0.5% of the original Unet architecture, manages a performance of around 0.255 on the testing dataset guided by the soft labels produced from the 4-Unet trained for 300,000 iterations. We also experimented with the KL-divergence loss function instead of the usual cross entropy function. However, this swap yielded no extra benefits (the best test loss remained around 0.25) and

Trial Number	1	2	3	4	5	6	7	8	9	10
2-Unet	0.156	0.142	1.371	0.138	1.376	1.379	1.380	1.381	0.223	1.384
1-Unet	1.37	1.35	1.36	1.33	1.38	1.38	1.36	1.33	1.30	1.31

Table 4. Reruns of the smaller unet architectures with the vanilla training procedure. The reported numbers are the average cross entropy losses calculated on the testing dataset with the parameters $\beta_t = 25$ and $n_t = 500$.

$T_{transfer}$	Test (avg.)	Test (min)	Training Loss (Hard)	Training Loss (Soft)
2	0.490	0.480	0.268	0.167
5	0.505	0.481	0.267	0.239
10	0.506	0.480	0.268	0.305
15	0.507	0.479	0.270	0.326
20	0.507	0.480	0.271	0.334

Table 5. Network performance of the 2-Unet guided by both soft and hard labels. The teacher model is 4-Unet trained for 300,000 iterations. The averaged test loss has the averaging parameters $\beta_t = 25$ and $n_t = 500$. The training soft losses are reported after multiplying by $T^2_{transfer}$.

hence, we continued with the cross entropy loss for the rest of the work. Here too, as mentioned in the above section, the convergence to 0.250 loss was much faster ($\tilde{1}$5x) with the teacher signal than without it (vanilla training).

3.7 Mixed Distillation

As outlined in [7], we use a training objective that combines the cross entropy loss between $\mathcal{P}^*_t$ and $\mathcal{P}^*$ and the hard loss between $\mathcal{P}^*$ and the actual binary labels. As described in section 3.3, the gradients from the soft loss are scaled by $1/T^2_{transfer}$ the soft loss needs to be multiplied by $T^2_{transfer}$ for the losses to be comparable in magnitude. Referring to Table 5, we observe that even though the mixed distillation procedure doesn't solve the 0.480 loss problem, it highlights the generalization issue present in the training process. Observe that even though the training losses are much lower (around 0.270), the network is not able to generalize it's learned knowledge to the test set. This hinted at using some sort of regularization method such as the Batch Normalization layer as described in Section 3.9.

3.8 Sequential Distillation: A failed attempt

In analogy of the parallel and sequential processes, we also try out the sequential version of the mixed distillation as described in the above section. In particular, we start with a smaller network (1-Unet) trained partially through the soft distillation procedure (Section 3.4). Now this partially trained network is further trained in the vanilla way using only the hard labels. Referring the results 6, we observe that such a procedure fails miserably and the testing loss blows very fast once the hard training of the pre-trained model is started. Although, this experiment clearly fails, it hints the hypothesis that the direction for optimizing the soft and the hard loss are very different in general and only when they are considered in parallel as in [7, 16] does the distillation procedure actually work. Also, remarkable is the rate at which the test loss blows up to much higher values than 0.480 when starting the hard training. Within a few hundreds of

Starting Model			Test Loss
Loss	#Iterations	$T_{transfer}$	
0.48	10,000	5	0.883
0.48	20,000	5	0.595
0.550	20,000	10	0.497
0.480	10,000	20	0.590
0.480	20,000	20	0.498

Table 6. Cross entropy loss for 1-Unet trained through sequential distillation. The averaging parameter for the test loss are $\beta_t = 25$ and $n_t = 1000$. The reported loss for the starting model is the test loss calculated using $\beta_t = 25$ and $n_t = 500$.

iterations, the testing cross entropy loss increases from 0.5 to over 1.20 for all the mentioned cases in Table 6.

3.9 Batch Normalization layer

As discussed in Section 3.7, we introduce the batch normalization layer [18] for introducting stronger regularization in the network. Although, this increases the number of learnable parameters by a small amount, the benefit in network performance far outweigh this minor drawback. We introduce the Batch Normalization layer in the expansive part of the architecture before the ReLU activation. Refer to Section 2.1 and Figure 1 for details.

We test the effect of introducing the batch normalization layer on the 4-Unet with vanilla training and find that it significantly **improves the network performance from 0.0930 to 0.0727** using the cross entropy loss as metric on the test set under identical averaging settings, $\beta_t = 25$ and $n_t = 500$. Referring to the loss curve in the figure 4, we report that this extra kick in network's performance comes at the expense of **a 5x longer training time** (from 300,000 iteration without the BN layer to around 1.5 million iterations with the BN layer).

Similar improvements of **around 22% in the cross entropy loss** are also achieved on the 16-Unet and the 32-Unet architectures. This makes a strong case for the use of the Batch Normalization Layer in all future uses of the architecture.

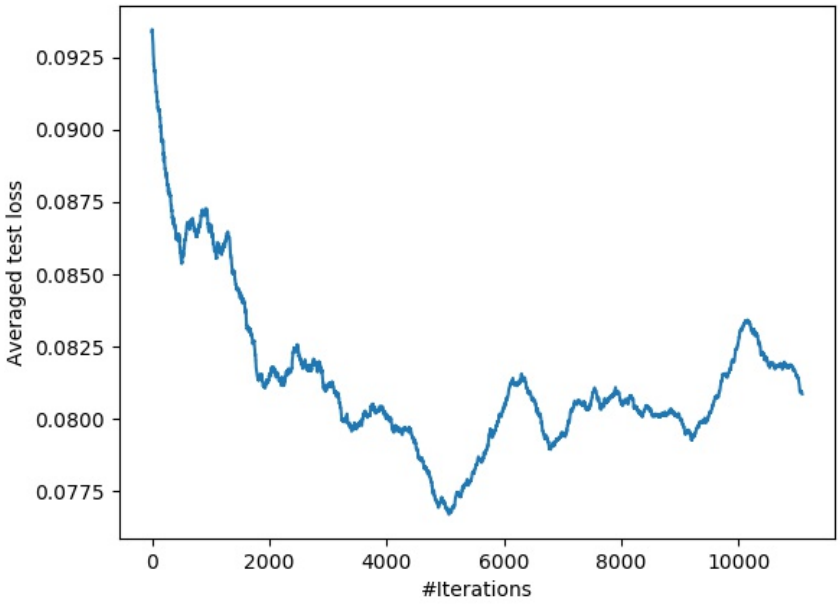

Fig. 4. The test loss curve for training the 4-Unet with the BN layer for 3.6 million iterations. The averaging setting for the reported curve is $\beta_t = 500$ and $n_t = 300$ to allow a smoother graph. The lowest test loss achieved is 0.07660 near 1,600,000 training iterations and takes around 2.2 days to train on a single Titan X GPU.

3.10 Making It Work: Batch Normalization & Class weights

Motivated from our experience from Section 3.6 - 3.9, we employ two techniques to overcome the 0.48 loss local minima. First, we introduce the Batch normalization layer in the U-Net (see Section 3.9) for both the teacher and the student model. This provides regularization to the network improving the generalization error as observed in Table 5. Secondly, we use class weights to penalize the network more on predicting wrong on the minority class as described in [15]. Combined, these two improvements enable distilling the 2-Unet to performance similar to 64-Unet. Moreover, with these enhancements the stochastic effects of initialization on the 2-Unet performance are also overcome and different reruns of the same experiments yield consistent results (in contrast to without distillation training as reported in Table 3).

Final Results Using only soft labels, U-net is trained to a cross entropy loss of 0.1340 (averaged over 3 trials) corresponding to an intersection over union score of 0.752 at $T_{transfer} = 5$. With same condition, but using both the soft and the

hard labels (Section 3.7), we could get to 0.135 cross entropy loss, equivalent to an Intersection over Union score of 0.759. Notably, while both the methods produce similar final results, the convergence rate with mixed distillation (∼100,000 iterations) is ∼1.5x faster than with only soft labels (∼155,000 iterations). To conclude, using the method described above, *a much smaller version of U-net (2-Unet with ∼1% of the original parameters) can be trained to perform similar to the original 64-Unet, both in terms of cross entropy loss (0.135 for 2-Unet vs. 0.102 for 64-Unet) and the intersection over union score (0.759 for 2-Unet vs. 0.804 for 64-Unet).*

4 Conclusion

We successfully demonstrate a procedure for compressing the U-net architecture **by over ∼100x** to just 1% of the original parameters (31,042,434 trainable parameters fo 64-Unet to 30,902 in 2-Unet) with minor decrease in performance, both in terms of cross entropy loss (0.135 for 2-Unet vs. 0.102 for 64-Unet) and the intersection over union score (0.759 for 2-Unet vs. 0.804 for 64-Unet). Additionally, we introduce the batch normalization layer in the architecture for regularization. We also discover several interesting patterns and make important observations that improve the architecture in other ways, some of which are briefly listed below.

- The U-net model can be significantly compressed without any extra effort! (Section 2.1)
- Challenges that data augmentation poses for Distillation. (Section 3.2)
- The stochastic performance degradation in Unet. (Section 3.5)
- Faster convergence rate with teacher guided training (Section 3.5)
- The dynamics of sequential and parallel guidance of soft and hard labels (Section 3.7/3.8)
- Performance boost from the Batch Normalization layer (Section 3.9)

Acknowledgement

I would like to thank **Dr. Mathieu Salzmaan** for mentoring me for the project, **Dr. Pablo Marquez Neila** for his valuable advice during key junctures and **Prof. Pascal Fua** for providing me with an opportunity to work on the project and granting the access to lab's computing resources. It has been a tremendous learning opportunity to take a closer look into the workings of deep learning. Thanks a lot!

References

1. Learning Efficient Object Detection Models with Knowledge Distillation, Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, Manmohan Chandraker, 2017
2. MobileID: Face Model Compression by Distilling Knowledge from Neurons, Ping Luo, Zhenyao Zhu, Ziwei Liu, Xiaogang Wang and Xiaoou Tang, 2016
3. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014)
4. Newell, Alejandro, Kaiyu Yang, and Jia Deng. "Stacked hourglass networks for human pose estimation." European Conference on Computer Vision. Springer International Publishing, 2016.
5. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2015.
6. Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006.
7. Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." arXiv preprint arXiv:1503.02531 (2015).
8. Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In European Conference on Computer Vision, pages 525–542. Springer, 2016.
9. Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. arXiv preprint arXiv:1605.04711, 2016.
10. Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In NIPS, volume 2, pages 598–605, 1989.
11. Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. NIPS, pages 1135–1143, 2015
12. Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2014). Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550.
13. Model Distillation with Knowledge Transfer from Face Classification to Alignment and Verification,Chong Wang, Xipeng Lan and Yangang Zhang, 2017
14. Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. International Conference on Learning Representations (ICLR), 2016
15. Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 234-241). Springer, Cham.
16. A. Lucchi, K. Smith, R. Achanta, G. Knott, P. Fua, Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks with Learned Shape Features, IEEE Transactions on Medical Imaging, Vol. 30, Nr. 11, October 2011.
17. C. J. Becker, K. Ali, G. Knott and P. Fua. Learning Context Cues for Synapse Segmentation, in IEEE Transactions on Medical Imaging, vol. 32, num. 10, p. 1864–1877, 2013.
18. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International Conference on Machine Learning. 2015.