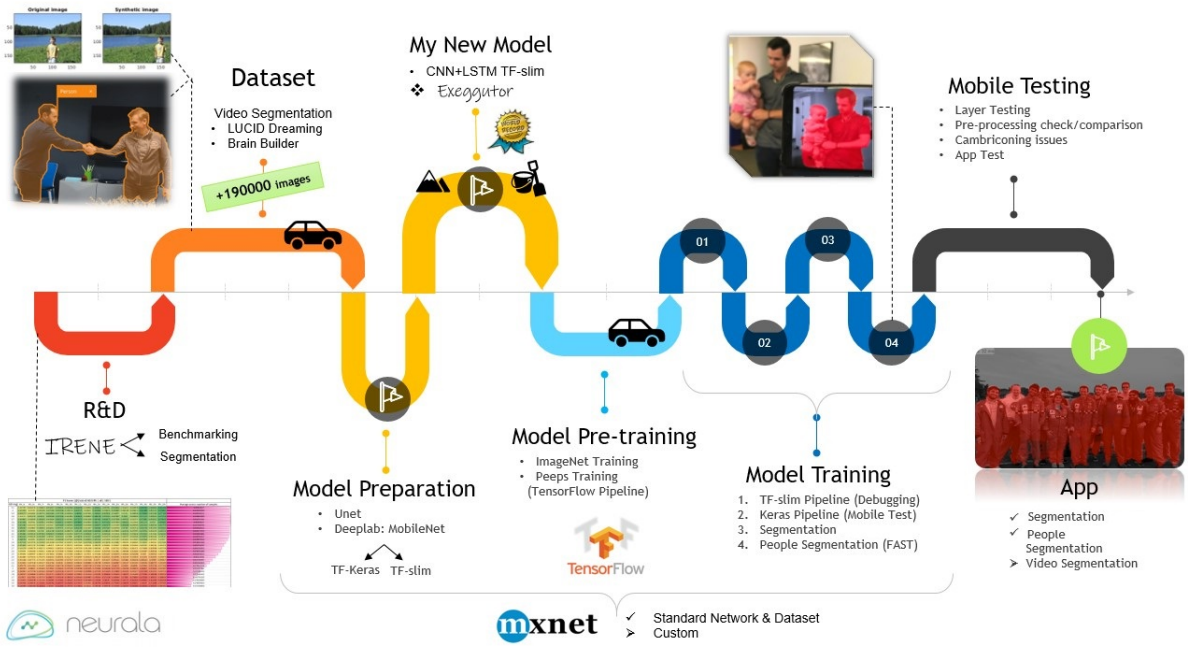
Learning to learn: A tale of data, neural models, and mobile implementations

Aishwary Jagetia
Worcester Polytechnic Institute
MAY-DEC 2018



***Abstract*—An overview of my work at Neurala.**

I. INTRODUCTION

From research to data creation, model preparation and training to mobile application deployment, my journey at Neurala is summarized in the above roadmap. Neurala is the artificial intelligence software company developing novel deep learning technologies and deploying neural network into smart artificial intelligence products like cameras, robots, drones, toys and self-driving cars more autonomous and interactive. It was a great opportunity to broaden my skill set by working on projects like classification, object detection, image segmentation, video segmentation, and their combinations on Tensorflow, Caffe and MXNet frameworks. Understanding and implementing techniques such as transfer learning, synthetic data creation, architecture developments, mobile deployment and novel classification techniques was a boon to me. Overall as a researcher it was an excellent exposure to the responsibilities one holds performing exciting tasks.

Advisors: (Neurala) CTO Dr.Anatoli Gorchet,
(WPI) Professor Dr.Carlo Pinciroli.

The purpose of this report is to summarize my contribution in the projects I have been a part of at Neurala. The outline of this report is project based, providing basic introduction and background of the project followed by the tasks performed along with their results.

II. PROJECTS

At Neurala I have been a part of 4 major projects with various tasks, which includes *Image Segmentation* for 10 classes (person, sky, plants, pet, building, flower, water, food, mountain beach), *Fast People Segmentation*, *Video Segmentation* and *Irene*. All of the projects other than *Irene* requires optimization based on mobile application, like inference time, size, memory etc. which has always been a sub-task throughout my internship. In this report I will discuss my accomplishments along with the challenges faced during the implementation.

A. *Irene*

Introduction:
Irene is a framework for object semantic segmentation

i.e. object detection with class classification and semantic segmentation which is based on SNIPER (Scale Normalization for Image Pyramids with Efficient Resampling) [1] adding multi-scale characteristics along with novel classification technique namely *NEMO* classifier which requires very few images for training purposes. Currently, Irene is in research phase with feasible results for object detection. In the recent past there have been an excellent work to bring in the consistency and more promising results.
My task was to understand the concept and the existing pipeline for Irene based on MXNet (python) and benchmark the object detection results on COCO dataset. Later integrate the segmentation head with a new in-house developed backbone. Also, perform various tests on non-COCO dataset to validate robustness of the framework.

Background:
All the way from Sliding-window detectors to SNIPER (Scale Normalization for Image Pyramids with Efficient Resampling) [1] there have been various small development in Object detection technique. Use of region proposal method to generate region of interest (ROIs) along with a feature extractor backbone in networks such as R-CNN [2], Fast R-CNN [3], Faster R-CNN [4] and R-FCN [5] are improvements based on the efforts to reduce the computational cost in object detection. In-addition to these effort R-FCN 3000 [6] was developed which uses superclasses and theoretically runs at 30fps makes it a viable option to be used within SNIPER [1]. Scale invariance was another problem which has been dealt with SNIP [7] without reducing the training samples. Combining all these efforts, SNIPER [1] helps in performing multi-scale training, which is based on Faster-RCNN with a ResNet-101 backbone giving better results on COCO dataset [18]. Use of Soft-NMS [9] within SNIPER [1] is an effort to reduce the false positives occurring for object detection, which can be applied across classes within single scale or across multi-scale output.
Mask-RCNN [10] is an old approach that does both object detection and instance segmentation but compared to SNIPER [1], Mask-RCNN [10] does not provide consistent features for NEMO classifier to classify object, which makes SNIPER [1] a better option to be incorporated within Irene.

Approach:
1) Task 1: Basic Understanding of the pipeline: After understanding the concept and the existing pipeline for Irene framework, I started observing the result of Irene over COCO training set. The results where consistent but not accurate. Based on the previous NEMO hyperparameters the large scale predictions were poorer, reason being the filtering of Soft-NMS across all 3 scales. After excluding Soft-NMS across 3 scales, we noticed improvements in 5K images. Tweaking the code a little we were also able to figure out the images performing poorly within the 5K validation images. It was also observed that the SNIPER weights are being loaded thrice which adds up training and inference time, which can be a potential surgery to reduce the inference time in the near future.

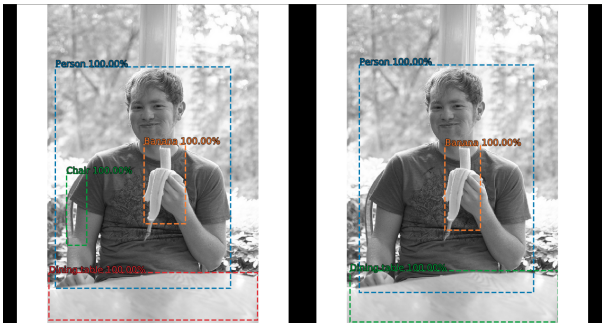


Fig. 1. Irene result (right) on a black and white image compared to ground truth (left)

2) Task 2: Benchmarking Irene: A novel training regime which I call as "Learning classifier dataset" suggested by Anatoli, is basically to train NEMO staring with a few number of samples per class and evaluating it on a random set of 100 images from the COCO training dataset. We then calculate the images with poor performance and select those for re-generating the classifier dataset to re-train NEMO. It is possible that the images selected may not include all classes and therefore we have selected additional images to have a balanced samples per class to be unbiased over a certain class. Re-training and evaluating NEMO for some iteration until the number of samples per class get saturated or is less than 20. Back then we were not able to save and load the weights for NEMO, due to which we had to re-train NEMO from scratch for all samples within the class. This training regime was time consuming also due to poor inference time.

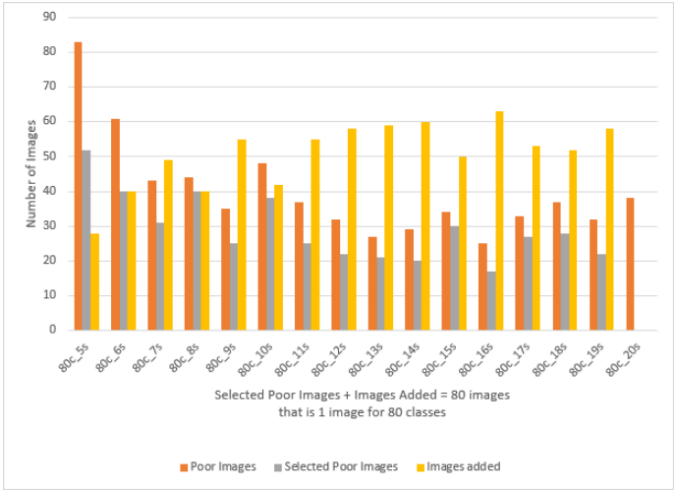


Fig. 2. Summary of how our training regime selected the images

We therefore add 1 sample for every class including those poor images in our classifier data and re-train it. With this regime we created 15 classifier datasets (5,6,...20). While comparing those dataset we found out that some images have a switch behavior (accuracy of one image increases while other decreases and vice-versa) which apparently was due to a NEMO hyperparameter. Therefore, we started gathering data for 100 randomly selected images between NEMO

hyperparameter (1.2 to 4.1) and samples performance (80 class 5 samples to 80 class 20 samples). The data gathered for 100 images are summarized as a heat map for F1 Score comparison.

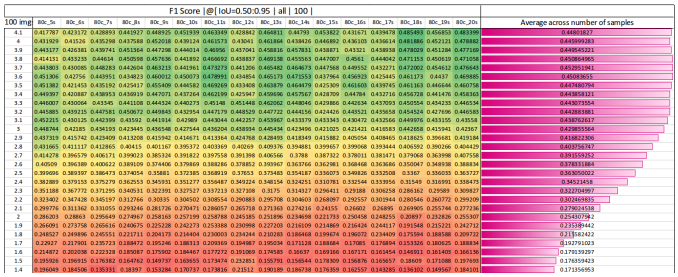


Fig. 3. Heat map for F1 Score comparison between Classifier dataset and NEMO hyperparameter

Based on these results 18 and 20 number of samples for 80 class are performing better with NEMO hyperparameter set to 3.7 as the optimum balance between Precision and Recall. Our learning regime provided us with better parameter values and therefore improved our results by a very small percentage as summarized below:

Irene Results on COCO 5K val

	IoU	area	maxDets	Previous results	New results
Average Precision (AP)	@[IoU=0.50:0.95	all	100	0.289	0.300
Average Precision (AP)	@[IoU=0.50	all	100	0.431	0.449
Average Precision (AP)	@[IoU=0.75	all	100	0.328	0.340
Average Precision (AP)	@[IoU=0.50:0.95	small	100	0.200	0.206
Average Precision (AP)	@[IoU=0.50:0.95	medium	100	0.350	0.356
Average Precision (AP)	@[IoU=0.50:0.95	large	100	0.396	0.416
Average Recall (AR)	@[IoU=0.50:0.95	all	1	0.292	0.317
Average Recall (AR)	@[IoU=0.50:0.95	all	10	0.463	0.495
Average Recall (AR)	@[IoU=0.50:0.95	all	100	0.477	0.511
Average Recall (AR)	@[IoU=0.50:0.95	small	100	0.337	0.358
Average Recall (AR)	@[IoU=0.50:0.95	medium	100	0.573	0.604
Average Recall (AR)	@[IoU=0.50:0.95	large	100	0.539	0.588

Fig. 4. Irene object detection results on COCO 5K validation images

There is room for further improvement in the performance of Irene, including faster inference may be with the help of new backbone or by profiling the pipeline. It will be also interesting to try to incorporate new loss function like Focal loss and compare the object detection results.

3) Task 3: Test Robustness: Amazon dataset contains some of the classes which are not part of COCO dataset on which Irene is pre-trained on. Irene performed very well on Amazon dataset as shown in figure 5. I also tried training the classifier with only 1 and 2 classes and found out the results for 1 class are very poor, which were due to NEMO hyperparameter as found out by Anatoli and Graham. From 2 class onwards the results were not affected by the specific NEMO hyperparameter.

4) Task 4: Integrating segmentation with new backbone: SNIPER does provide with untrained segmentation head, but to get the weights we were required to perform the entire training procedure (training backbone with Image and then entire

Fig. 5. Irene results on Amazon dataset shows robustness of the framework

network with COCO dataset) which required MXNet pipeline support. Therefore, with the help of GluonCV a toolkit integrated within MXNet I was able to set up an entire pipeline for out training purpose. Multi-processing bug along with opencv installation issue was a hurdle however, after resolving the bug, everything installed properly. The pipeline is ready and we can train, test and validate a Classification, Detection, Segmentation (Semantic and Instance) on standard networks (ResNet's, MobileNet's, VGG's, SqueezeNet's, DenseNet's, AlexNet, darknet, Inception, SENet) with standard dataset (COCO, ADE20K, PASCAL VOC, Cityscapes, ImageNet, CIFAR). It also runs on multi-gpu, multi-core, with an ability to export weights (Json, Param, txt files) from Python to be used on C++ and vice versa. To validate the pipeline I trained InceptionV3 and CIFAR and achieved the published results. I also tried using the SNIPER training pipeline which had a lot of bugs to begin with, whereas the new MXNet pipeline based on GluonCV worked exceptionally well.
Having the pipeline set up the next task was to train the new backbone with MXNet pipeline. The in-house developed backbone was written in C++ and experiments were performed on Caffe framework. Therefore, I had to re-write the entire architecture in Python API within the MXNet framework. To test if the pipeline is performing equally well as compared to Caffe framework, I trained the network "SigNet-1024ss" (developed by Anatoli) on CIFAR dataset. The following is the convergence plot of "SigNet-1024ss" training.

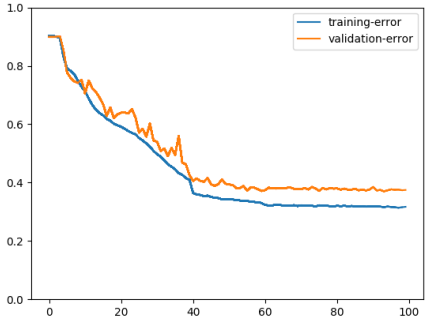


Fig. 6. initial results of "SigNet-1024ss" trained on CIFAR dataset

These results very much dependent on the initial learning rate and weight decay. Although, with a single cascade the val-

idation accuracy was 80.5% achieved 66% validation accuracy which can be improved further.

B. Video Segmentation

Introduction:
Video Segmentation refers to analyzing video images and segmenting them with the help of temporal information from the previous frames. Although, there are various techniques and networks published in the DAVIS video object segmentation not all of them are feasible for mobile application. Also, mobile compatibility makes it a very challenging project.
As a team our task was to understand various existing techniques and come up with the most feasible one which needs to be deployed onto the device. This may require re-writing an entire new network or coming up with a new technique inspired from the existing one. Also, this project had a requirement of sequential dataset for people video segmentation.

Background:
Most of the existing techniques such as OSVOS [11], OnAVOS [12] (Adaptive OSVOS), MaskTrack [13], FAVOS [16], Lucid Dreaming [17] are based on the fine-tuning of the first frame segmentation mask. Although, OSVOS and OnAVOS are based on heavy networks which make them difficult to be deployed on a device, the use of thin-plate-splin method as proposed by Jeff makes MaskTrack a viable candidate to be deployable within devices. FAVOS take the first mask and creates 300+ bounding boxes that have high overlap with the mask which help in tracking the movement of each box in the next frame and simultaneously performing segmentation within the box. Lucid Dreaming uses the first frame and the mask and generates a sequence of images along with the mask, but it requires heavy transformation and MATLAB implementation. On the other hand STFCN [15] is a techniques which take time-distribution into consideration for temporal information. It is based on LSTM layers placed at the bottom of UNet [20].

Approach:
1) Task 1: Device compatible techniques development: FAVOS (Fast and Accurate Online Video Object Segmentation via tracking Parts) claims to run faster than OSVOS, OnAVOS and LUCID Draming techniques make a viable option to be used for mobile deployment. Unfortunately, my effort to run Caffe based FAVOS in Deeplab was unsuccessful. It was also challenging to implement STFCN in a mobile device, because of the device compiler issue which creates multiple model segments. I successfully created a Dummy CNN-LSTM model without using Time-Distribution layer in Keras TF framework, which is first of it's kind. Fortunately, the new DDK support multiple segments now making STFCN a viable option. It will be interesting to see how STFCN (CNN-LSTM) works for video segmentation.

2) Task 2: Dataset creation: We collected dataset from various online sources like DAVIS, YouTube dataset and also generated a significant data using in-house video annotation tool called as "Brain Builder". However, without the use of L-DNN it is better to have as big as possible dataset and therefore, we used the existing technique Lucid Dreaming to satisfy our data need. Lucid Dreaming is written in MATLAB which

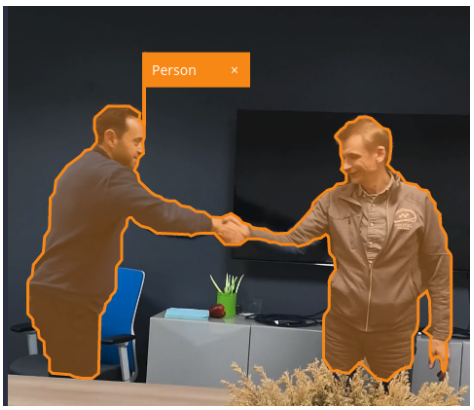


Fig. 7. Data produced using Brain Builder Annotation tool

is based on patch-filling and regeneration of the random (non-sequential) output. I wrote a pipeline to generate 20 sequential images using Lucid Dreaming with MATLAB implementation using the curated Peeps dataset as our seed images.

Fig. 8. Lucid Data Dreaming results

The MATLAB implementation is a very slow process involving 300 iterations to patch-fill every mask. With the help of profiler I was able to improve the inference speed to a good extent. As a result, I was able to generate 196680 images from 9834 seed images and use it for our video segmentation project. The data is sequential it can be used for normal people image segmentation as well as for pre-training purposes.

C. *Image Segmentation*

Introduction and Background: Image Segmentation is process of partitioning a digital image into multiple segments. understanding the existing networks like UNet [20], ResNet [21], DensNet [22], Deeply Supervised CNN [23], SE Block [24],etc. were very useful in designing new networks.

Approach:

1) Task 1: Model Preparation: I started helping my team mates to re-create a tensorflow model which we inferred to provide us with the most promising solution. I tested and debugged TF-slim MobileNet model and also worked on UNet model. I wrote the TF-Keras version of MobileNet which proved out to be the key to make it compatible with the device compiler (a tensorflow model) and deploy it onto the device.
During this phase I learned to train a dummy model and save it's weights. Understanding the weights flow within a existing network can help everyone in designing a new network. Also, I learned that with deep network the data requirements increases. I helped in debugging the TF-Slim pipeline for batch size and model segment issue.

2) Task 2: Model Development: Taking inspiration from UNet and Deeply Supervised CNN networks, and based on the prior knowledge, I made an attempt to create a new model. There are two version called Exeggcute and Exeggutor. The difference is that Exeggutor uses Depthwise Convolution Block within the Size specific Block (as shown in figure 11) and Exeggcute uses Convolution Block replacing Depthwise Convolution Block. It is important to note that both networks consider the direct input image features in the downsampling path.

Exeggutor is 5.5 MB in weights which goes down to 4.4 MB after compiling. It takes 25.56 MB memory and runs at 45-51 ms of inference speed. Whereas Exeggcute is 3.3 MB in weights which goes down to 2.9 MB after compiling. It takes 21.29 MB memory and runs at 31-46 ms of inference speed. Both of these networks runs on the old DDK, also I was able to test run Exeggcute on New DDK as well.

It will be interesting to further try following modification to improve upon the results with these architectures:

- Replace the depthwise with resnet/SE/Dense/Deep layer aggregation block and check the change in inference and memory.

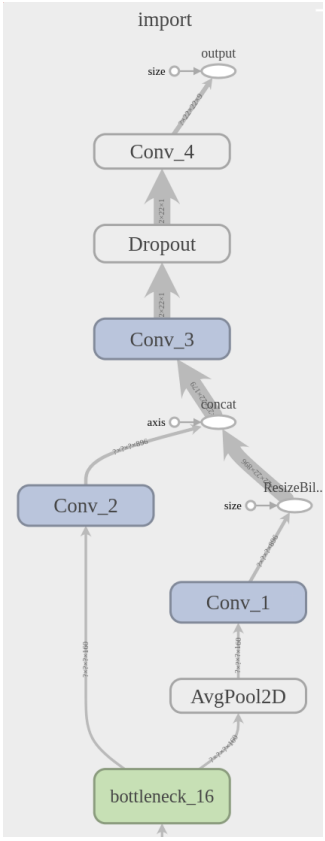


Fig. 9. Visualizing weights flow through a network. Broader the width of the flow shows the higher importance of the path

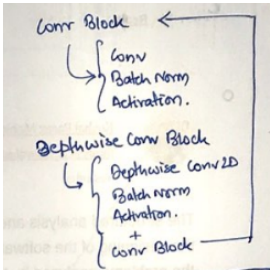


Fig. 10. Legends for Conv Block and Depthwise Block

- Train it on ImageNet for downsampling, then on COCO-VOC, COCO-stuff and Peeps (added 4% accuracy) dataset for the upsampling (upto 352) and chop off the last resize layer.
- Have variable learning rates within the layers while training.
- Try out "HE" or "MSRA" initialization with RELU.
- Try out 1x9 and 9x1 filters w/wo separable convolutions.
- Crop images of 352 size instead of resizing them as input images.

.

3) Task 3: Model Pre-Training: Transfer learning is an important step required to have higher accuracy not only for classification purpose but also for Segmentation and

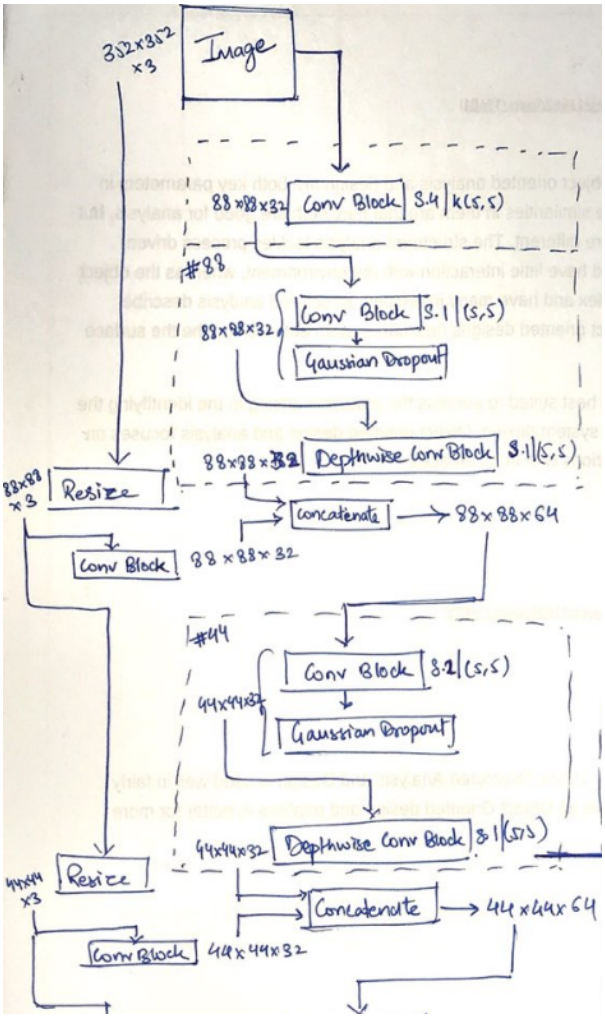


Fig. 11. Exeggutor downsampling path

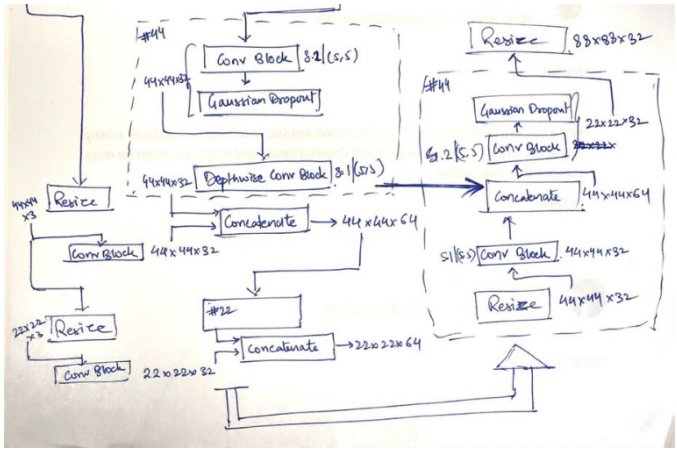


Fig. 12. Exeggutor down-up U connection

Object Detection. I have developed and tested ImageNet classification python based pipeline for any tensorflow models.

4) Task 4: Model Training: During this project I modified and trained several model with keras pipline. Also, individually worked on updating the brains with faster networks with different input and output sizes to improve resolution for Neurala customers.

Model			Accuracy		mIOU	
Name	Specs/Dataset	Class	Pixel Acc	Mean Acc	w (bg)	w/o (bg)
Exeggutor	1st balanced dataset without any pre-training (Imagenet / Peeps)	10	81.69%	83.25%	70.05%	70.83%
	1st balanced dataset with Peeps pre-training	10	82.36%	82.90%	71.26%	72.06%
	depth layers replaced by conv & 1st balanced dataset without any pre-training (Imagenet / Peeps)	10	82.28%	82.73%	71.09%	72.04%
	New July 31 dataset for 8 class (softmax) (tested on Curated Test Set)	8	88.09%	89.79%	79.32%	80.33%

Model		IOUs											
Specs/Dataset	Class	bg	person	sky	plant	death	pet	building	flower	water	food	mountain	beach
1	10	62.31%	64.22%	91.79%	60.05%	-	75.60%	72.25%	74.69%	75.84%	68.83%	56.71%	68.27%
2	10	63.24%	68.36%	92.77%	57.98%	-	77.27%	74.79%	75.66%	77.34%	69.98%	57.32%	69.16%
3	10	61.63%	63.84%	91.93%	58.47%	-	76.75%	72.95%	75.18%	77.21%	74.30%	57.39%	72.32%
4	8	71.17%	83.18%	91.87%	67.84%	-	81.99%	88.43%	82.20%	82.64%	64.52%		

Fig. 13. Exeggutor results

D. Mobile testing and App Deployment

Introduction:

Deploying a neural model in a device requires optimization in memory, size and inference which are the most important and challenging factors to be considered. At Neurala we use two neural processing systems from leading hardware manufacturers [25] to make our model deployable on the device, satisfying customer requirements.

My task was to test different layers within a network along with solving bugs and complete a processing comparison between PC and device outputs. Along with performing App tests and solving device compiler issue, I also worked on the Android side to have better understanding for model creation.

Approach:

1) Task 1: Layer Testing Android App Test: Running a single layer dummy network through a single image from both PC and Mobile and visualizing the difference between the outputs helped me perform a sanity check for individual layers. I analyzed the float 16 and float 32 difference between PC and mobile devices along with a RGB and BRG test. Also, not all the layer are device compilable which should be taken into consideration while designing a network. For example, using tensorflow slim dropout layer or tensorflow slim batchnorm layer creates multiple model segments and will not work on the device if the weights are not saved properly.

It was also important to understand how different layers affect the size, memory and inference on the device. It was challenging to solve the blowup issue (the size of the device compiler output was larger than the model weights) in one of the Deeplab model with depthwise layer, during the implementation of Image Segmentation project.

2) Task 2: Running comparisons: It is important to analyze the visual outputs from the device after-all these efforts are to make the output user friendly.

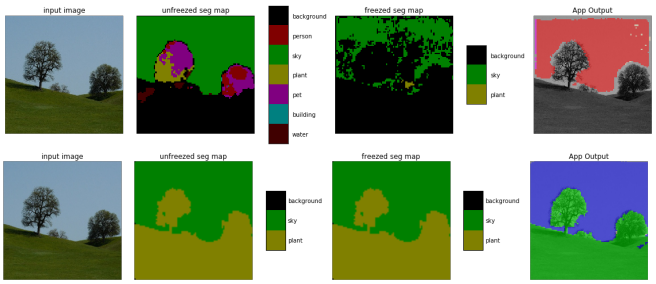


Fig. 14. Comparison of the model performance on PC and on Mobile

3) Task 3: Solving device compiler issue : When we started the tensorflow side of image segmentation all the initial test which were performed gave equally good results irrespective of the tensorflow model being deployed, which made me realize that there is a loading issue wherein only caffe models are loaded along with a model initialization error for all tensorflow models. It is important to take into consideration that different device compiler techniques requires different parameter while training (for example the scaling factor). Additionally, it is always a very good approach to re-start any test with a dummy model in difficult situations.

III. Conclusion

A exciting work exposure at Neurala was a learning experience for me. Although, the frustration with installation of version specific software's did make my life difficult for couple of weeks, it also taught me to be persistent. Every task and project I did at Neurala upskilled myself. Working for Brain Builder Marketing or being a part of Neurala Research Meeting and Reading Club among the eminent researchers in AI, have been an excellent education for me. This experience will be a foundation to all my future endeavours.

IV. Feedback

Anatoli Gorchet: (Co-Founder and CTO)
"Courage and desire to try new things. Sherlock Holmes was so successful primarily because he had a wast array of past cases to draw upon, this will come with experience naturally, just keep doing and learning. I'd recommend you to go beyond backpropagation in neural network theory. I think the main reason for miscommunications we had came from me not explaining clearly enough and you not asking clarifying questions. Ask more! Good production outcome! "

Abhishek Gaur: (Deep Learning Engineer)
"Keen learner, patient, and asks good questions. Wanting to do more than is possible in the given time frame (not particularly a bad thing). Just need a bit more organizational skills! I'd recommend you to try and focus on getting a bit more organized. I'm super happy with how you did during the entire co-op so just knit picking, no major flaw that I noticed. Overall Motivated individual and great team player."

Matt Luciw: (Research Scientist)
"I believe you're competent, interested in and sometimes having fun with the work, energetic, quick to volunteer to do necessary tasks. I'd recommend you to work on organizing/leading your own projects because you'll have to do that someday, probably. Over the internship your communication skills got a lot better, I would recommend to keep working on it. Overall It was a Great job. It's been a pleasure."

Jeff Rodny: (Senior Research Scientist)
"Very good internship! Sometimes we forget you're not full time here. Quick learner, open to new ideas, ability to apply oneself."

Jeremy Wurbs: (Deep Learning Researcher)
"I think you work very well in a team, coordinating with others to accomplish a larger task. I think coming into the projects you did, which were quite complex and storied, can be quite difficult, but I think you've done quite well to catch on quickly and started to be productive after a very short period of time for each project. There is a world of knowledge to know, especially in deep learning. But I think it's a new field that we're all trying to learn, so we'll all be learning for some time to come. Programming skills, paper reading skills, presentation skills, team leading skills – they're all useful and development naturally. It was always very easy and pleasant to work with you, and working on larger goals seemed to break into individual tasks smoothly and efficiently."

Santiago Olivera: (Lead CI Engineer)
"Good at making polls and fancy GUI/slides. I'd recommend going deep enough to be able to customize dev tools. Keep doing what you are doing!"

Aditya Gupta: (Deep Learning Engineer)
"Very persistent and keen on learning. I'd recommend to analyze the problem with more depth before coming to conclusion. I would love to have you as a team mate, it's always very fun to discuss things with you."

Michael Bishop: (Senior Software Engineer)
"Ability to solve problems. I very much enjoyed working with you!"

Crystal Typermass: (AI Marketing Director)
"Execution - you do what you say, very hard worker, humble, kind, a pleasure to work with. Willingness to learn, great attitude, in addition to being a fantastic engineer - great marketing person too."

Vesa Tormanen: (VP of Engineering)
"Very energetic and flexible!"

Daniel Glasser: (VP of Client Operations)
"You show diligence in completing work, you don't let ambiguity or uncertainty stop you from getting things done. From the perspective of a project manager, communication

is key. Progress updates, summaries, details about what is accomplished and what work remains are always helpful. I see you have a strong work ethic and have demonstrated your ability to contribute to team efforts in a meaningful way. My only suggestion would be that as you progress in your career, look for opportunities to take the lead and set the pace for others around you."

REFERENCES

[1] Singh, B., Najibi, M. and Davis, L.S., 2018. SNIPER: Efficient Multi-Scale Training. arXiv preprint arXiv:1805.09300.

[2] Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[3] Girshick, R., 2015. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[4] Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[5] Dai, J., Li, Y., He, K. and Sun, J., 2016. R-fcn: Object detection via region-based fully convolutional networks. In Advances in neural information processing systems (pp. 379-387).

[6] Singh, B., Li, H., Sharma, A. and Davis, L.S., 2018. R-FCN-3000 at 30fps: Decoupling Detection and Classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1081-1090).

[7] Singh, B. and Davis, L.S., 2018. An Analysis of Scale Invariance in Object DetectionSNIP. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3578-3587).

[8] Hosang, J.H., Benenson, R. and Schiele, B., 2017, July. Learning non-maximum suppression. In CVPR (pp. 6469-6477).

[9] Bodla, N., Singh, B., Chellappa, R. and Davis, L.S., 2017, October. Soft-nmsimproving object detection with one line of code. In Computer Vision (ICCV), 2017 IEEE International Conference on (pp. 5562-5570). IEEE.

[10] He, K., Gkioxari, G., Dollr, P. and Girshick, R., 2017, October. Mask r-cnn. In Computer Vision (ICCV), 2017 IEEE International Conference on (pp. 2980-2988). IEEE.

[11] Caelles, S., Maninis, K.K., Pont-Tuset, J., Leal-Taix, L., Cremers, D. and Van Gool, L., 2017. One-shot video object segmentation. In CVPR 2017. IEEE.

[12] Voigtlaender, P. and Leibe, B., 2017. Online adaptation of convolutional neural networks for video object segmentation. arXiv preprint arXiv:1706.09364.

[13] Perazzi, F., Khoreva, A., Benenson, R., Schiele, B. and Sorkine-Hornung, A., 2017, July. Learning video object segmentation from static images. In Computer Vision and Pattern Recognition (Vol. 2, No. 7).

[14] Hu, Y.T., Huang, J.B. and Schwing, A., 2017. Maskrnn: Instance level video object segmentation. In Advances in Neural Information Processing Systems (pp. 325-334).

[15] Fayyaz, M., Saffar, M.H., Sabokrou, M., Fathy, M., Klette, R. and Huang, F., 2016. STFCN: spatio-temporal FCN for semantic video segmentation. arXiv preprint arXiv:1608.05971.

[16] Cheng, J., Tsai, Y.H., Hung, W.C., Wang, S. and Yang, M.H., 2018. Fast and Accurate Online Video Object Segmentation via Tracking Parts. arXiv preprint arXiv:1806.02323.

[17] Khoreva, A., Benenson, R., Ilg, E., Brox, T. and Schiele, B., 2017. Lucid Data Dreaming for Multiple Object Tracking. arXiv preprint arXiv:1703.09554.

[18] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollr, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

[19] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). Ieee.

[20] Ronneberger, O., Fischer, P. and 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.

[21] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[22] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., 2017, July. Densely connected convolutional networks. In CVPR (Vol. 1, No. 2, p. 3).

[23] Zhu, Q., Du, B., Turkbey, B., Choyke, P.L. and Yan, P., 2017, May. Deeply-supervised CNN for prostate segmentation. In Neural Networks (IJCNN), 2017 International Joint Conference on (pp. 178-184). IEEE.

[24] Hu, J., Shen, L. and Sun, G., 2017. Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507, 7.

[25] Ignatov, A., Timofte, R., Szczepaniak, P., Chou, W., Wang, K., Wu, M., Hartley, T. and Van Gool, L., 2018. Ai benchmark: Running deep neural networks on android smartphones. arXiv preprint arXiv:1810.01109.