Capstone Project

April 21, 2018

1 Machine Learning Engineer Nanodegree

1.1 Capstone Project

David Eraso
April 17th, 2018

1.2 I. Definition

1.2.1 Project Overview

This project is defined by the Kaggle/Booz Allen Hamilton competition 2018 Data Scince Bowl in which the objective is to find cells' nuclei in divergent images to advance medical discovery.

Identifying the cells' nuclei allows researchers to identify each individual cell in a sample, and by measuring how cells react to various treatments, the researcher can understand the underlying biological processes at work. By automating nucleus detection, research could unlock cures faster by allowing more efficient drug testing.

In an image segmentation task since we need to divide the image into segments, or regions, belonging to the same object. To achieve this it's necessary to classify each pixel in the image into one out of a set of predefined classes. This is also called a dense classification problem. Such an output yields a segmentation of the image because nearby pixels are often of the same class and hence, such classification tends to segment the input.

In the case of image masking there are two classes: we are concerned with separating a relevant image with background. For the specific problem at hand, we are interested in nucleus detection; therefore, this is an image segmentation problem in which we want to classify pixels as belonging to a nucleus, or as background. Each image usually contains several nuclei.

Datasets are obtained from the Data Web Page of the competition.

The dataset contains segmented nuclei images acquired with diverse imaging modalities and containing different cell types. Each image has its associated masks for each nucleus in the image. Since masks are not allowed to overlap no pixel belongs to two masks. The datasets used in this project belong to the training and testing sets of stage 1 of the competition.

1.2.2 Problem Statement

The objective is to "create a computer model that can identify a range of nuclei across varied conditions". The evaluation metric that will be used is the mean average precision (mAP) at different Intersection over Union (IoU) thresholds.

This is an Image Segmentation problem: the images have overlapping objects, nuclei, and we want to correctly identify the object and its boundaries. The mask is a binary classifier that defines whether or not a given pixel is part of a given nucleus or not (background or other nuclei).

We start with a banchmark model relevant to the problem at hand. We then browse over 8 other candidate architectures for semantic segmentation and implement other two architectures inspired on the key ideas of two of these arcitectures. Finally we'll try to improve the score of the best alternative.

1.2.3 Metrics

From the 'Evaluation' link on the competition webpage:

This competition is evaluated on the mean average precision at different intersection over union (IoU) thresholds. The IoU of a proposed set of object pixels and a set of true object pixels is calculated as:

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

The metric sweeps over a range of IoU thresholds, at each point calculating an average precision value. The threshold values range from 0.5 to 0.95 with a step size of 0.05.

In other words, at a threshold of 0.5, a predicted object is considered a "hit" if its intersection over union with a ground truth object is greater than 0.5.

At each threshold value t, a precision value is calculated based on the number of true positives (TP), false negatives (FN), and false positives (FP) resulting from comparing the predicted object to all ground truth objects:

$$\frac{TP(t)}{TP(t) + FP(t) + FN(t)}$$

A true positive is counted when a single predicted object matches a ground truth object with an IoU above the threshold. A false positive indicates a predicted object had no associated ground truth object. A false negative indicates a ground truth object had no associated predicted object. The average precision of a single image is then calculated as the mean of the above precision values at each IoU threshold:

$$\frac{1}{|thresholds|} \sum_t \frac{TP(t)}{TP(t) + FP(t) + FN(t)}$$

Lastly, the score returned by the competition metric is the mean taken over the individual average precisions of each image in the test dataset.

Justification:

The mAP at IOU is the most common accuracy metric when dealing in general with obect detection that uses bounding boxes as labels. These boxes are hand labeled boxes put over the image to specify where in the image the object of interest is located. In the case of semantic segmentation, instead of boxes, masks are used to identify the object and the classification is made at pixel level.

For the specific problem at hand, precision is better since the number of negative samples is very large. This is an imbalanced dataset in the sense that negative sample pixels (background, cell soma, and anything not considered part of a nucleus in the image) are larger than positive samples. Since the precision denominator contains the false positives, precision is not affected by a large number of negative samples, that's because it measures the number of true positives out of the samples predicted as positives (TP+FP). Precision is more focused in the positive class than in the negative class, it actually measures the probability of correct detection of positive values,

while other metrics such as FPR and TPR (ROC metrics) measure the ability to distinguish between classes.

1.3 II. Analysis

1.3.1 Data Exploration

Images contain a variety of cell types and imaging techniques; thus, the algorithm must have the ability to generalize. Each Image has its corresponding `ImageId`, this is the name of the folder containing the files belonging to that image. For the training set, this folder contains two subfolders; the `images` subfolder with the image file, and the `masks` subfolder which contains the segmented mask for *each nucleus*. So there is one mask file for each nucleus. *Masks are not allowed to overlap, i.e. no pixel belongs to two masks*. The test set will only contain the `images` subfolder.

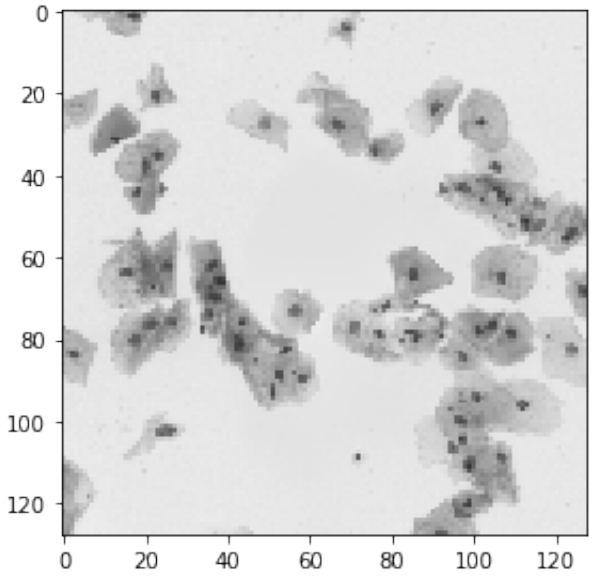
Using as a reference the jupyter notebook `Nuclei_finder.ipynb`:

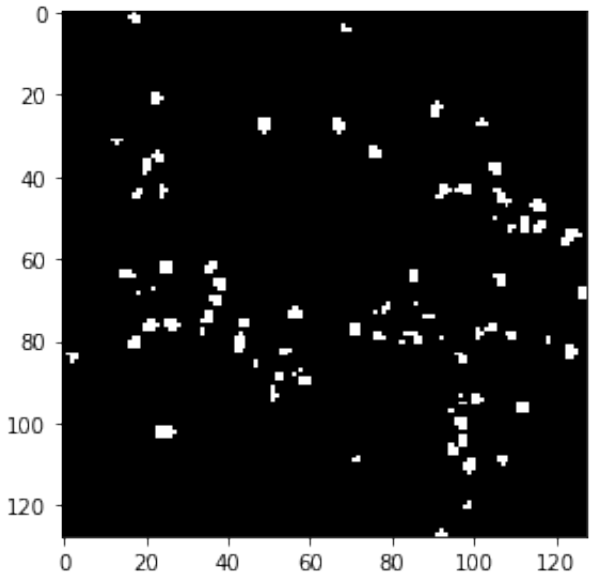
```
Number of training ids: 670
Number of test ids: 65
```

1.3.2 Exploratory Visualization

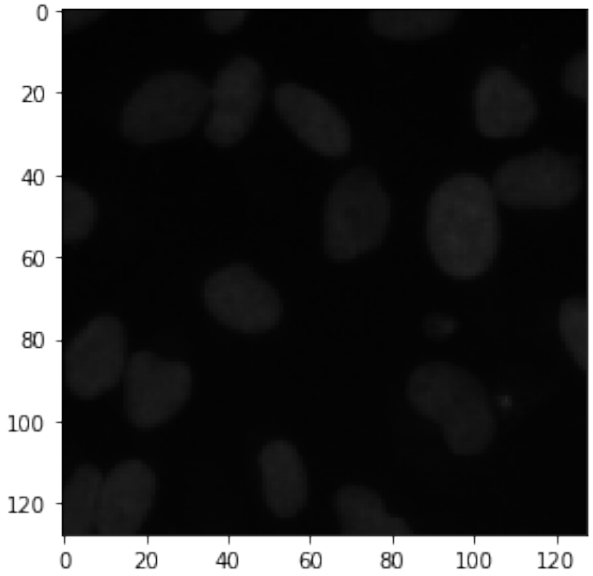
Again, using as a reference the jupyter notebook `Nuclei_finder.ipynb`:

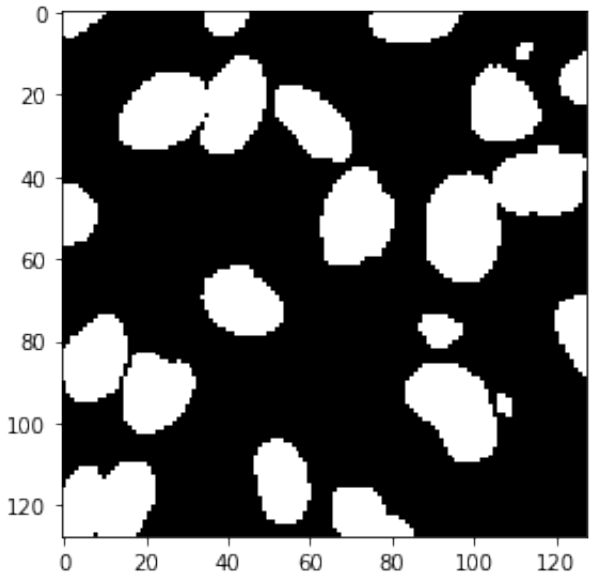
Images and masks are fetched and resized to image parameters of image width and height of 128, and 3 channels. This is a sample of a training image with its respective masks in one image:





In this particular pair, cells are well defined and nuclei are small darker spots as revealed by the masks.





In this other pair, only nuclei are visible as lighter larger spots.

As mentioned before, images in this dataset are diverse. Also, as with any human-annotated dataset, there may be various forms of errors in the data. Also, masks are not allowed to overlap.

1.3.3 Algorithms and Techniques

Beyond the benchmark model, discussed in the next subsection, there is a list of architectures for Semantic Segmentation which were roughly examined to implement the main idea behind two of these. Along with the benchmark model there will be three architectures that will be scored. The architectures for Semantic Segmentaion are:

1. Fully Convolutional Networks for Semantic Segmentation Ref: Semantic Segmentation using Fully Convolutional Networks over the years

In the paper the authors state that they "adapt contemporary classification networks (AlexNet, the VGG net, and GoogLeNet) into fully convolutional networks and transfer their learned representations by fine-tuning to the segmentation task."

Basically, the authors convert existing classification ConvNets to fully convolutional nets by replacing the last fully connected layers of the classsification net with convolutions; in the authors words: "the fully connected layers can also be viewed as convolutions with kernels that cover their entire input regions". For pixelwise prediction, instead of a softmax classification layer at the end, there is a deconvolutional process back to a 'heatmap' of pixels of the original image.

This architecture relies upon existing classification ConvNets. Usually these ConvNets are built and tuned for images with several objects, depth and in general rich and diverse in features.

The original paper dates back to 2014. So the state of the art in image semantic segmentation has evolved since then.

2. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation The Encoder-Decoder architecture proposed in the SegNet design uses an encoder network identical to the 13 convolutional layers of the VGG16 network, and a decoder network that maps the low resolution encoder feature output to full input resolution feature maps for pixel-wise classification.

According to the authors; "the decoder uses pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear upsampling". Reusing these max-pooling in-

dices in the decoding process, among other practical advantages, improves boundary delineation. Also, SegNet is more memory efficient than FCN

This architecture was primarily motivated by "scene understanding applications". Compared to other architectures this one hasn't been a top performer.

3. Multi-Scale Context Aggregation by Dilated Convolutions Dilated Convolutions begins with the premise that even though previous semantic segmentation models are based on adaptations of ConvNets originally designed for classification, semantic segmentation problems are structurally different. The authors present a module that uses dilated convolutions "based on the fact that dilated convolutions support exponential expansion of the receptive field without loss of resolution or coverage".

In classification, pooling reduces the dimensionality of the input (down-sampling) allowing generalization and thus translation invariance; however, this reduces resolution, which is undesirable for segmentation. The dilated convolution layer uses a filter that covers a larger area (preserving resolution), while only taking the same amount of inputs as a regular pooling by introducing gaps in the kernel.

4. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs This architecture makes use of the previous idea calling the dilated convolution an "Atrous" convolution. Its latest version is Rethinking Atrous Convolution for Semantic Image Segmentation. Initially, Atrous Spatial Pyramidal PRefineNet "exploits all the information available along the down-sampling process to enable high-resolution prediction using long-range residual connections. In this way, the deeper layers that capture high-level semantic features can be directly refined using fine-grained features from earlier convolutions". A new component called "chained residual pooling", used to capture background context efficiently, is deployed.

Standard FCN suffer from downscaling of features maps losing resolution. Dilated convolutions fix this problem but are computationally expensive. RefineNet attempts to make use of various levels of detailat different levels of convolutions joining these details to obtain high resolution without the need of large intermediate feature maps.

This architecture seems promising for the problem at hand. Let's implement a second model roughly inspired on RefineNet:ooling (ASPP) is introduced. ASPP is basically multiple parallel atrous convolutional layers with different sampling rates. ASPPs are the key contribution, but the paper also states that "the commonly deployed combination of max-pooling and downsampling in DCNNs achieves invariance but has a toll on localization accuracy"; therefore, the responses at the final DCNN layer are combined with a fully connected Conditional Random Field (CRF).

5. RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation RefineNet "exploits all the information available along the down-sampling process to enable high-resolution prediction using long-range residual connections. In this way, the deeper layers that capture high-level semantic features can be directly refined using fine-grained features from earlier convolutions". A new component called "chained residual pooling", used to capture background context efficiently, is deployed.

Standard FCN suffer from downscaling of features maps losing resolution. Dilated convolutions fix this problem but are computationally expensive. RefineNet attempts to make use of various levels of detailat different levels of convolutions joining these details to obtain high resolution without the need of large intermediate feature maps.

This architecture seems promising for the problem at hand. *Te second model is roughly inspired on RefineNet.*

6. Pyramid Scene Parsing Network PSPNet is concerned with scene parsing; i.e. prediction of the label, location, as well as shape for each element. Dilated convousions are used and the authors

propose a pyramid pooling module to aggregate the context by applying large kernel pooling layers. The complete architecture is as follows: first a ConvNet yields a feature map, then "a pyramid parsing module is applied to harvest different sub-region representations", upsampling and concatenation layers come next to form the final feature representation, and finally a lasst convolution layer outputs the final per-pixel prediction. There is also an "auxiliary loss" applied before the input to the pyramid pooling module.

The third model is roughly inspired on PSPNet using transfer learning.

7. Large Kernel Matters -- Improve Semantic Segmentation by Global Convolutional Network Large Kernel Matters proposes that a large kernel size is desirable in dense classification problems such as semantic segmentation. Since larger kernels are computationally expensive, the k x k convolution is approximated with a sum of 1 x k + k x 1 and k x 1 and 1 x k convolutions, and the authors call this a "Global Convolutional Network" that addresses both classification and localization issues. They also suggest a "residual-based boundary refinement to further refine the object boundaries".

8. Rethinking Atrous Convolution for Semantic Image Segmentation As mentioned before, this is the latest version of the architecture mentioned above. ASPP is improved with a 1x1 convolution and three 3x3 atrous convolutions with different rates. There is also the use of batch normalization.

1.3.4 Benchmark

The benchmark model is the kernel `U-Net starter - LB 0.277`, by Kjetil Åmdal-SævikKeras. Beyond implementing the basic setup of getting the data as well as encoding results for submission, Kjetil builds a U-Net model, *'loosely based on U-Net: Convolutional Networks for Biomedical Image Segmentation and very similar to this repo from the Kaggle Ultrasound Nerve Segmentation competition.'*

This model scored 0.261 in the leaderboard. The author reported a score of 0.277.

This benchmark uses the Keras functional API which is a useful way for characterizing complex architectures such as the U-Net.

In the original paper; U-Net: Convolutional Networks for Biomedical Image Segmentation, the training makes "strong use" of data augmentation. Also, in the benchmark model architecture, although the same number of levels is used (number of convolutional layers as well as number of maxpooling and transpose convolutions in both the contracting path used to capture context as well as the symmetric expanding path used for localization) the main difference with the original architecture resides on the number of filters that are used in the convolutionals layers:

In the original architecture: 64, 128, 256, 512, 1024, 512, 256, 128, 64

In this benchmark model: 16, 32, 64, 128, 256, 128, 64, 32, 16

To change the number of filters back to the original values increases significatively the number of parameters from 1,941,105 to 31,031,745 parameters. This architecture was tried (`my_nuclei_finder.ipynb`) without showing improvement (there was no submission for this model, there was no improvement in validation loss and early stopping kicked in after a `patience` of 10). This means that there may be other features of the architecture that couls improve performance.

1.4 III. Methodology

1.4.1 Data Preprocessing

All the data preprocessing is taken from the benchmark kernel. Image parameters are set to `IMG_WIDTH` = 128, `IMG_HEIGHT` = 128, `IMG_CHANNELS` = 3 for both the training and testing images and 1 channel for the masks. The images are specified to be of type numpy `uint8` and the masks numpy `bool`.

For the benchmark model, U-Net inspired, and the RefineNet inspired model, there is a lambda layer right after the input layer that normalizes the input image to have values between 0 and 1. The PSPNet inspired model uses transfer learning and images are fed into the first layers of a VGG19 network. The transfer learning process will be explained in detail when the implementation of this architecture is described.

1.4.2 Implementaion

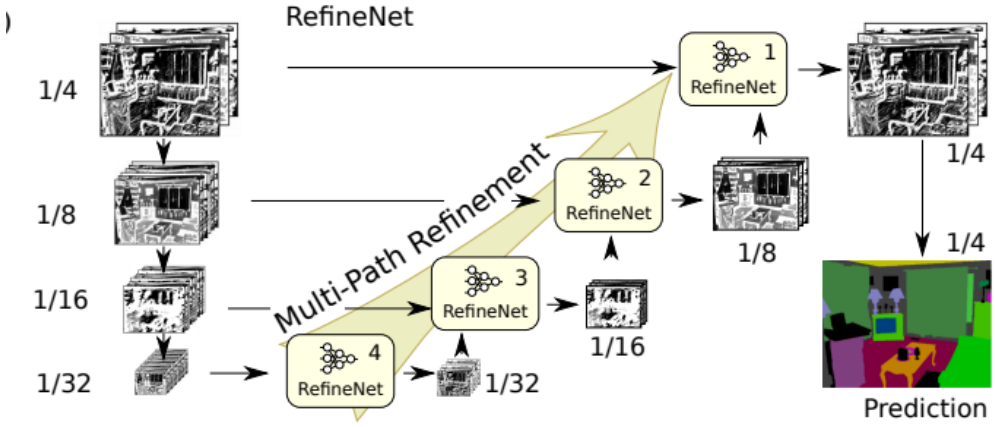
Implemetation process

The models were implemented using Keras Functional API, which is a useful way for defining complex models, that are not sequential and which is the case for these implementations. The internal `mean_iou` metric makes use of tensorflow's metric. For model fitting batch size is 16 (the final model uses baatch size 32), and epochs is set to 50 making use of `EarlyStopping` and `ModelCheckpoint` to avoid unnecesary training and keeping track of the best model.

The most challenging coding was the use of transfer learning with only the first few layers of a pre-trained model (see 'PSPNet inspired model using transfer learning'), and data augmentation for a dataset with masks as labels (see 'Refinement').

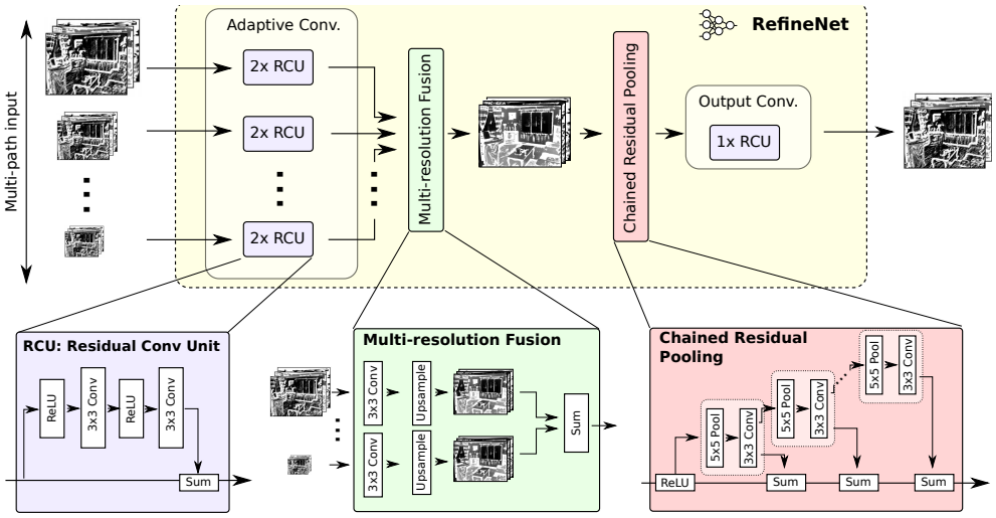
1. RefineNet inspired model

Initially, the contracting path is similar to the one in the benchmark model, only one level shorter in order to fit 3 RefineNets modules in total. The original paper proposes 4 RefineNets modules in the expanding path as such:



From RefinNet paper

Each RefinNet module is composed of a Residual Conv Unit (RCU) block, a Multi-resolution Fusion block, and a Chained Residual Pooling block as such:



From RefinNet paper

To simplify further the architecture in the implementation, not only the RefinNets modules are reduced from 4 to 3; in the RCU block of each module there is only one convolution layer instead of 2. Also, the pair pooling-convolution in the Chained Residual Pooling block is kept at the minimum of 2.

Despite these simplifications, this second model scored 0.283. That is an improvement of 8.4% over the first score using the benchmark. Even though the RefineNet inspired architecture is one level shallower than the benchmark, the RefineNet modules make this last architecture more complex: The RefineNet inspired architecture has a total number of parameters of 3,525,313, while the benchmark model (U-Net inspired) has 1,941,105 total parameters. So the number of parameters increase 81,6%.

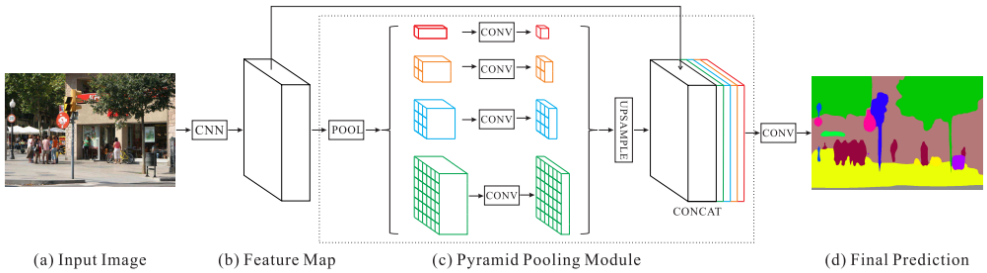
2. PSPNet inspired model using transfer learning

This architecture allows for transfer learning to be used more naturally. Models with corresponding pre-trained weights available in Keras Applications are pre-trained on ImageNet. Since we have a small dataset very different from the original dataset, in order to make use of Transfer Learning we would like ideally to only use the features extracted on the very earliest layers. For this we have to make use of the h5 file of the pre-trained weights of the model we will use as base. The PSPNet module can the be added on top of the earliest layers of the base model.

The base model that is used is the VGG19 Network. Initally the `summary` for this model is examined, as well as the layer names for the h5 pre-trained weights file that was downloaded. Only the two first blocks of this architecture are used since we are interested to use only the most basic image features. Each block is assembled of two 3x3 convolutional layers and one 2x2 pooling layer. So a base model is built with these characteristics in which the VGG19 weights are loaded. When the final model is put together, the layers of the base model will be freezed by setting their `trainable` attribute to False.

The output for the base model has dimensions (None, 32, 32, 128), before adding the PSP module, an expanding path is added. This is a two level expanding path using transpose convolutions. It also makes use of concatenation as in the U-Net making use outputs in the contracting path. This yields the feature map used as input in the PSPNet module.

The Pyramid Scene Parsing module has the following form:



From PSPNet paper

Since the PSP module implementation is not costly in terms of complexity, this implementaion was as similar as possible to the original

This model was fitted twice; the weights/submission-df `model-dsbowl2018-3.h5/nuclei_finder_PSPNet_deraso.csv` scored 0.278, and the weights/submission-df `model-dsbowl2018-3v2.h5/nuclei_finder_PSPNetV2_deraso.csv` scored 0.287. They are both the exact same model. The first result was close to, but not as high as the score from the RefinNet inspired model (0.283), but the second fit was higher. This model that uses both transfer learning and the idea of PSPNet has a total number of parameters of 584,929, from which 324,769 are trainable parameters, making it the architecture with the least number of parameters.

Something to take into consideration: there is no evidence that choosing VGG19 as a base model for the transfer learning may be the best fit for this particular task. However, all architectures in Keras Applications are pre-trained with the ImageNet dataset and that's why only very few of the initial layers are used.

1.4.3 Refinement

Taking into account the previous discussion, implemetations and results, the model chosen to try to improve it looking for a better score was the PSPNet inspired model that uses transfer learning. Even though the RefineNet inspired model also achieved a score that was better than the benchmark and very similar to the PSPNet inspired model, the PSPNet inspired model uses a lot less parameters.

In order to improve its performance two changes to the original architecture were done:

1. Data augmentation.
2. Batch normalization

Data augmentation was by far the most challenging part. These dataset uses as 'labels' masks, which are 1 channel images. Thus, using `fit_genrator` after augmentation was not a possibility since this method assumes labels for classification. To use the regular `fit` method for training the network, `ImageDataGenerator` had to be used for creating an augmented training set and an augmented validation set. Further, the datastructure containing both the training and validation data/labels pairs had to be constructued by hand, doing this resulted on nested lists with the shape (`augmented_length`, 1, 128, 128, 3) that had to be corrected to (`augmented_length`, 128, 128, 3).

Batch normalization was implemented by simply adding `BatchNormalization()` layers after the convolutional layers of the original architecture except in the PSP module itself.

The final result is a score of 0.277, which is not an improvement. This means that in order to improve the generalization capability of a semantic segmentation network for this particular

imagery, beyond finding a better tuned architecture and finding a better way to implement data augmentation, there must be other approaches to correct errors; these will be discussed futher in the 'Improvement' section.

1.5 IV. Results

1.5.1 Model Evaluation and Validation

For this project the different models were evaluated and validated both by the internal metrics of training such as validation loss and the use of the mean of intersecrion over union metric from tensorflow `tf.metrics.mean_iou`, and the leaderboard competition scoring that uses the whole testing dataset. The results are as follow:

Model	Loss	Mean IOU	Validation Loss	Competition Score
U-Net arch.	0.0755	0.8028	0.05972	0.261
RefineNet arch.	0.0779	0.8122	0.05997	0.283
PSPNet arch. 1	0.0699	0.8335	0.05426	0.278
PSPNet arch. final	0.0648	0.8699	0.06048	0.277

U-Net arch. is the benchmark.
PSPNet arch. 1 with transfer learning *before* data augmentation, and batch normalization.
PSPNet arch. final with transfer learning *after* data augmentation, and batch normalization.

1.5.2 Justification

The two alternative architectures that were tried had an improvement from the benchmark. The RefinNet architecture had an improvement of 8.4% over the benchmark. The best score of the PSPNet, which is not in the main jupyter notebook, but is the exact same architecture, had a score in the leadrboard of the competition of 0.287. This is an improvement over the benchmark of 10.0%. The improvement on the reported score of 0.278 is of 6.5%.

To verify the robustness of the best option (PSPNet inspired model) the same architecture was fitted twice, these are the results:

Model	Loss	Mean IOU	Validation Loss	Competition Score
PSPNet arch. 1	0.0699	0.8335	0.05426	0.278
PSPNet arch. 2	0.0697	0.8399	0.05236	0.287
----MEAN----	0.0698	0.8367	0.05331	0.283
-VARIANCE-	0.00000002	0.00002048	0.000001805	0.0000405

Variance is negligible, showing consistency and robustness. The average for the Competition Score is the same as for the RefineNet arch. using 6 times less parameters.

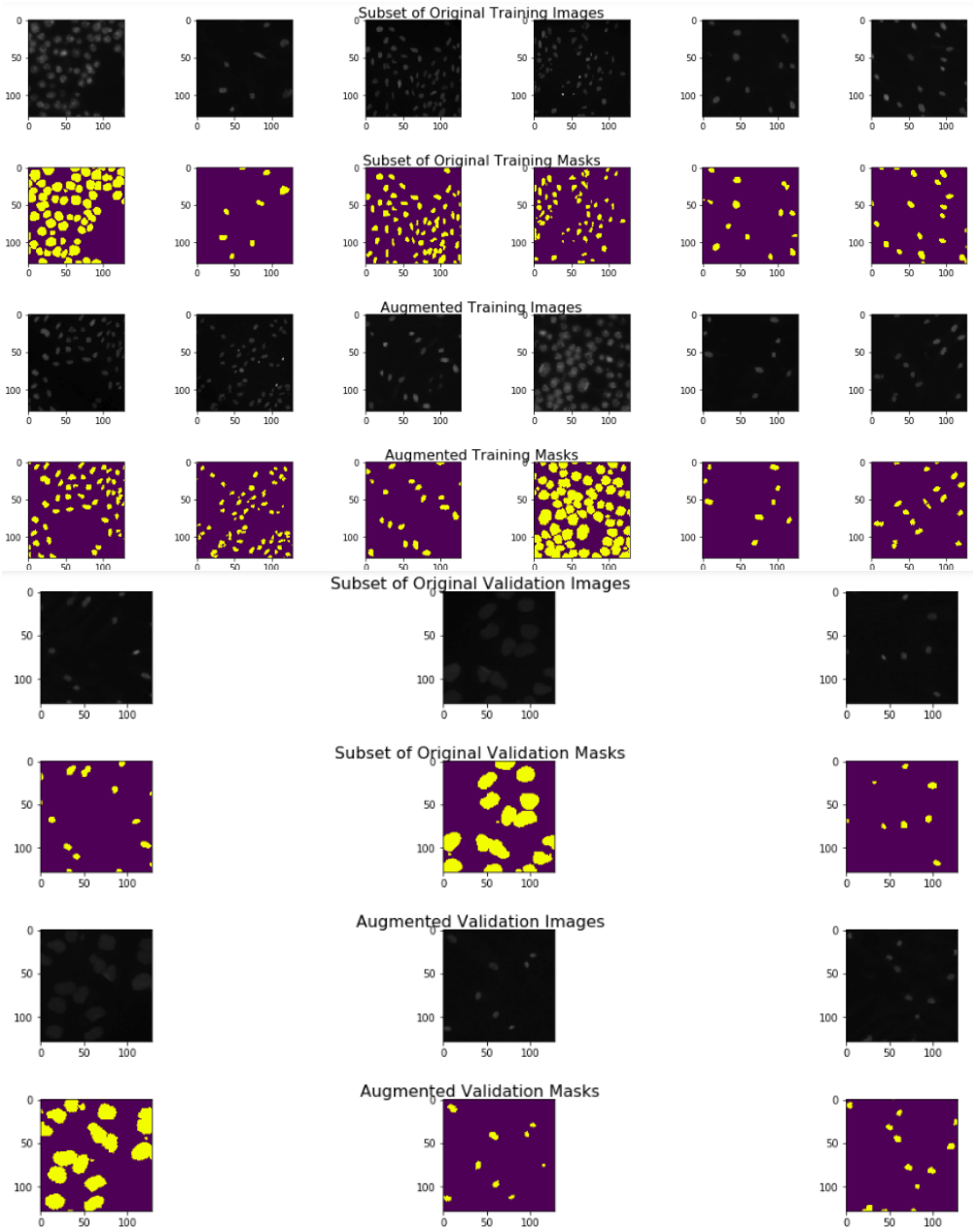
1.6 V. Conclusions

1.6.1 Data Augmentation Visualization

For data augmentation, given the nature of the images, the parameters used in `ImageDataGenerator` were as follow:

`rotation_range`=90; up to 90 degrees for random rotations. `width_shift_range`=0.5; up to 50% of the imagfe for random horizontal shifts. `height_shift_range`=0.5; up to 50% of the imagfe for random vertical shifts. `fill_mode`='reflect'; points outside the boundaries of the image are filled in the reflection mode. `horizontal_flip`=True; randomly flip inputs horizontally. `vertical_flip`=True; randomly flip inputs vertically.

Arguments such as featurewise center/std normalization, ZCA, shear, zoom, rescale were not included. The visulaization to check the augmented images in both training and validation pairs follows:



1.6.2 Reflection

This was a very hard problem to tackle. The state of the art for this kind of task despite significant advances in other fields of image computing is still deficient. Several ideas used for semantic

segmentation were implemented and there was a relatively better performance from a benchmark that is already a state of the art architecture for segmentation in medical images.

In this project, greater importance is conferred to exploring the state of the art in the realm of semantic segmentation in general. The top performers in the leaderboard focused on very heavy duty refinement of U-Net (the benchmark) that is discussed in the next section. However, in this project there is a good review and two interesting alternative implementations.

The most interesting aspects in the project for my learning journey are the handcrafted implementation of both transfer learning for only a few initial layers, and data augmentation for masks labels. Also is very importatn to mention the personal implementations using Keras functional API of complex blocks such as the RefineNet block and the PSPNet block, this gives me the confidence to implement networks with my own code of interesting ideas found in the literature.

1.6.3 Improvement

An interesting idea put forward in the 'Discussion' section of the competition, by the leader score (0.631) uses U-Net with the following adjustments:

- 2 channels masks for networks with sigmoid activation i.e. (mask - border, border) or 3 channels masks for networks with softmax activation i.e. (mask - border, border , 1 - mask - border)
- 2 channels full masks i.e. (mask, border)

Also, use of very heavy data augmentation; and use ensembling by averaging masks before postprocessing. The Loss Function for networks with sigmoid activation and 2 channel masks they used combination of binary_crossentropy with soft_dice per channel; for networks with softmax activation and 3 channel masks they used combination of categorical_crossentropy with soft_dice per channel (soft dice was applied only to mask and border channels).