TableResNet - Giving TableNet a New Backbone

Brian Rhindress
CS231n
brhindre@stanford.edu

Abstract

In this project, I replicate and extend TableNet, a recent, high-performing multi-task table detection model. Extracting tabular information from document images is important to millions of organizations, globally. TableNet frames the multi-task of semantically segmenting document images into table masks (table detection) and column masks(table structure recognition). This project builds on the open source TableNet Pytorch implementation by Tomas Sosorio (2021).[12] The key innovation in this project is replacement of the standard TableNet VGG19 encoder backbone with a ResNext encoder backbone. It is found that ResNext performs slightly worse than the VGG19 implementation after 2000 iterations of training. Additionally, saliency map visualizations show the ResNext TableNet version to predict slightly more erratically, and to focus less well on tabular sections of documents. Future work may include augmenting this dataset with more negative examples and further investigating alternative backbone issues as an optimization challenge.

1. Introduction

Nearly every organization can benefit from inferring tabular data from their paper or pdf document records. In particular, public organizations are often mandated to maintain long-term paper records, while ideally duplicating data in a centralized warehouse. There is a robust community of researchers working on this problem, and the International Conference on Document Analysis and Recognition (ICDAR) 2021 conference includes a competition on Scientific Table Image Recognition to LaTex.

There are two primary computer vision tasks regarding tables in images: table detection and table structure recognition. Table detection is the task of identifying if/where tables exist in an image. Table structure recognitions requires detecting the column and/or row boundaries within a detected table. Both tasks involve segmentation. This segmentation can be performed in several ways – either at the pixel level (semantic segmentation) or using bounding boxes to identify separate potential tables/columns (instance segmentation). Since the broader goal is to extract tabular information, these two primary segmentation tasks would typically be followed by additional post-processing steps, such as cell-structure identification, and Optical Character Recognition (OCR) to extract the actual alphanumeric content within the table.

In this project, I replicate and extend TableNet, a recent, high-performing multi-task table detection model.[6] TableNet joins a growing number of models that address the table detection and structure recognition tasks in a single end-to-end approach. The input to TableNet is an image from a word or Latex document that contains one or more tables. The model then encodes the image with a VGG19-based backbone, followed by two decoding branches for table and column detection. The outputs are table and column mask images, which are the same size as the original input image. While the original TableNet paper additionally uses rule-based semantic extraction for rows to complete the table structure, I only address table and column mask detection in this project.

2. Related Work

There is much recent and ongoing work in the field of table detection and extraction that largely fall into two categories: semantic segmentation and instance segmentation. Additionally, many papers in 2019 and 2020 contribute new data or augmentations to common datasets, as labeled tabular data is only now becoming widely available.

Recent developments in document-based semantic segmentation include efforts by Sarkar et al. (2019)[10] in extracting document structure. By first dividing each image into horizontal strips and predicting class participation, this model then leverages the prediction of the past row as the prior for future prediction. Thus, segmentation here is coupled with use of the hierarchical structure of the document. TableNet [6] takes a more straightforward approach, predicting table and column inclusion by pixel, after transforming document images with a shared pre-trained VGG19[11] backbone and two separate CNN branches for table and column prediction (Paliwal, et al., 2020). This simple approach

yields precision and recall scores in the range of 0.96 on the ICDAR 2013 and Marmot Table datasets. Finally, while it is not purely a table detection task, it is worth noting Chargrid (Reisswig, Katti, Höhne, et al., 2019),[9] which proposed a two-dimensional character grid representation of document images, as another recent and notable contribution to semantic segmentation-based document extraction.

Recent developments in instance segmentation include CascadeTabNet, which approaches the two-task table and cell segmentation problems with a 3-stage Cascade Mask R-CNN HRNet model.[7] The team utilized and adopted the terminology of the Mmdetection framework,[2] starting with a pre-trained CNN HR_NetV2p_W32 backbone. One of the great benefits of CascadeTabNet is a true end-to-end system, as the table and cell segmentation tasks are followed with rules, tailored to bordered vs. borderless tables and text extraction using Tesseract. (Prasad, et al., 2020). CascadeTabNet scored third in ICDAR 2019 and scored best on the ICDAR 2013 dataset with precision and recall of 1.0. Other instance segmentation approaches include use of Faster R-CNNs by Luo, et al (2021) [5] to extract data from financial documents and Mask R-CNN by Agarwal, Mondal, and Jawahar (2020).[1] Lastly, Raja, Mondal, and Jawahar (2020)[8] approach table detection and structure recognition from the opposite end of the process, detecting cells and nearby interactions to predict row and column associations with other cells. Each of these models typically use one of the publicly available datasets such as ICDAR-2013, ICDAR-2017, ICDAR-2019, UNLV, Marmot, PubLayNet, TableBank. TableBank is one of the more promising recent datasets, as it contains 417k high-quality labeled tables, ready for learning and evaluation. (Li, et al. 2019)[4]

Overall, while instance segmentation models such as CascadeTabNet are currently performing slightly better, there is appeal in the simplicity of a model like TableNet. Thus, due to reproducibility and comparably high performance, I decided to focus my project on training a TableNet model and extending it, where possible.

3. Methods

3.1. Approach

My approach was to first replicate the VGG19-based TableNet architecture, using the open source Pytorch implementation by Tomas Sosorio [12] and instructions from the original paper. Then, I tested replacing the VGG19 backbone with a ResNext50_32x4d backbone, with the intuition that a deeper network might more expressively detect tables and columns.

3.2. Original VGG19 Implementation

The original TableNet paper begins by preprocessing a document image by converting it to RGB and resizing to 1024x1024 pixels. Tomas Sosorio's Pytorch implementation uses 896x896 pixels, which I kept in the model. The essence of TableNet is an encoder-decoder architecture, with the encoder as a VGG19 backbone, followed by two decoder branches – one for table detection and one for column detection, respectively. The encoder backbone uses the pre-trained VGG19 model hosted by Pytorch (20,024,384 parameters costing 76MB). As a reminder, the key innovation of VGG was to use smaller, 3x3 filters with deeper networks for improved image classification. VGG19 includes 19 convolutional layers, divided into 5 groups by maxpooling layers, and followed by three fully-connected layers, ultimately ending with softmax and classification scores. For TableNet, only these first 5 sequential layers of CNNs and max pooling are kept. Images coming out of the VGG19 encoder are of size 512 x 28 x 28. The first step of the TableNet decoder has two shared 1x1 convolutional layers (C=512), which function to learn some foundational features of tables and columns. These layers, named "Conv6 + Dropout" in the original paper are shared by both the subsequent table and column decoder branches. The table decoder branch takes the output of Conv6 and passes it through a table-specific "Conv7" layer, another 1x1 convolutional layer. Then, the table-branch proceeds by upscaling the output with fractionally strided convolutions and concatenating it with the Max Pool4 layer (512 x 56 x 56) from the VG199 backbone. This type of step is repeated, again upscaling and concatenating with the Max Pool3 layer (256 x 112 x 112) from the VG199 backbone. Finally, this output is upscaled and converted from 1258 to 1 channel, which represent the predicted table mask scores (1 x 896 x 896), which are then passed through a sigmoid function. Similarly, after Conv6, the column decoder consists of two additional 1x1 convolutional layers called Conv7 and another 1x1 convolutional layer, Conv8. Just as in the table branch, this output is upscaled and concatenated with Max Pool 4, followed by upscaling and concatenation with Max Pool 3. Thus, the column decoder branch also concludes with an imaged-sized column mask (1 x 896 x 896). The key architectural insights are 1) pretrained VGG19 layers can be repurposed for table detection, 2) the table and column decoders should share some layers to capture similarity, 3) the table and column decoders should have some unique layers to capture the unique aspects of table and column detection, and 4) the table and column decoders should leverage some earlier, lower-level features learned by the finetuned VGG19 backbone. The common semantic segmentation loss function used by Tomas Sosario is "Dice Loss," which is defined as,

$$\frac{2 * \sum_{i,j}(scores_{i,j} * mask_{i,j})}{(\sum_{i,j}(scores_{i,j} + mask_{i,j}))}$$

which varies between 0 and 1.

3.3. ResNext Implementation

As a novel contribution, I swapped in a ResNext50_32x4d backbone [14] in place of the original VGG19 pretrained layers. See figure 1 for annotations in red, displaying my additions to the original model. The intuition behind this follows that of ResNet and ResNext, namely that deeper networks can result in improved performance. ResNet was initially successful in creating extremely deep networks that could be trained without vanishing gradients by including residual blocks to learn linear combinations of the identity function and activations.[3] However, after initial attempts at implementing ResNet-152, the model was too big (60,192,808 parameters costing 230MB) for my AWS allotment. Thus, I pivoted to ResNext, an architecture that combines ideas of residual blocks from ResNet as well as repeated topological blocks inspired by Inception modules.[13] ResNext allows for models of performance similar or better than ResNet, but with far fewer parameters by designing blocks of higher cardinality, defined by the size of the set of transformations. Thus, to save space and increase the expressivity of the original TableNet architecture, I used the pretrained ResNext50_32x4d model from Pytorch, which includes 50 layers of 32-cardinality (size: 25,028,904 parameters, 95.5MB). As notated on figure 1, in order to similarly pull lower-level features from ResNext and concatenate with the decoder branches, as was done with the VGG architecture, earlier activations in the ResNext pipeline needed to be reduced in the channel dimension. Sequential layers 2 and 3 were obtained from the ResNext backbone. To step the channel dimension down, rather than using pooling, 1x1 convolutions were used to convert ResNext sequential layer 2 from 512 to 256 channels x 112 x 112 and ResNext sequential layer 3 from 1024 to 512 channels x 56 x 56. Other than this change, the rest of the standard TableNet architecture remained the same in this version. One notable difference between my approach and that of the original TableNet paper is the original made updates by gradient descent in a 2:1 table:column ratio for the first stage of training, followed by 1:1 in a later stage. This implementation uses a 1:1 ratio consistently.

4. Dataset

TableNet trained on ICDAR-13 and the Marmot table dataset. For simplification, I follow the approach of Tomas Sosario and only use the Marmot table dataset for training and testing. The dataset includes 509 English document images, which begin at resolution 793 x 1123 pixels. I utilize the transformations in the starter code which resize each image to 1024 x 1024, then take a middle crop of the image at 896 x 896 pixels. Each document image includes a 1-channel table mask (tables are represented with a white box). Column masks were hand labeled by the TableNet team, so each image has both a 1-D table mask and a column mask. Table and column masks also began as 793 x 1123 resolution and were converted to 896 x 896 pixels using the same transformation step as the document images.

Regarding training/validation/test splits, the data was split into 85% training images (432) and 15% validation images (77). Validation hold-outs were considered sufficient for experimental purposes, so no additional test images were held out.

5. Experiments/Results/Discussion

5.1. Metrics

While the original TableNet paper used Precision, Recall, and F1, for practical implementation reasons, I here focus on Precision and Accuracy. This is imperfect, as Recall would be a better metric to check that the full tables are being realized, but accuracy and precision are sufficient for now to assess the performance of the model. These metrics are computed as:

$$Accuracy = \frac{\sum_{i,j}(scores_{i,j} > cutoff * mask_{i,j})}{(\sum_{i,j} mask_{i,j}))}$$

$$Precision = \frac{\sum_{i,j}(scores_{i,j} > cutoff * mask_{i,j})}{(\sum_{i,j}(scores_{i,j} > cutoff)}$$

5.2. Hyperparameters

I follow Tomas Sosario in implementing the same hyperparameters as the TableNet paper, using Adam optimization with a learning rate of 1e-4, beta1=0.9, beta2=0.999 and epsilon=1e-8, without weight decay. Batch for VGG batch sizes were 2 (as in TableNet) and for ResNext, batch sizes were 1, as required due to space constraints. Models were trained for over 2000 epochs each. While the original TableNet paper used 0.99 as the cutoff for table prediction, I test prediction with a 0.5 and 0.90 cutoff. The difference between which these cutoffs is found to be negligible.

5.3. Results

Overall, three main model specifications were tested: the original TableNet specification with VGG19, TableNet with VGG19 and batch normalization, and the novel ResNext contribution of this project. Each model was trained for over 2000 epochs. Table and Column accuracies and precision were plotted for both training and validation sets as well as loss functions.

As shown in figure 3, the loss profile for both tables and columns have very similar shapes, likely owing to the fact that most layers (thus, predictive tendencies) are shared between tables and columns. When smoothed, each of the

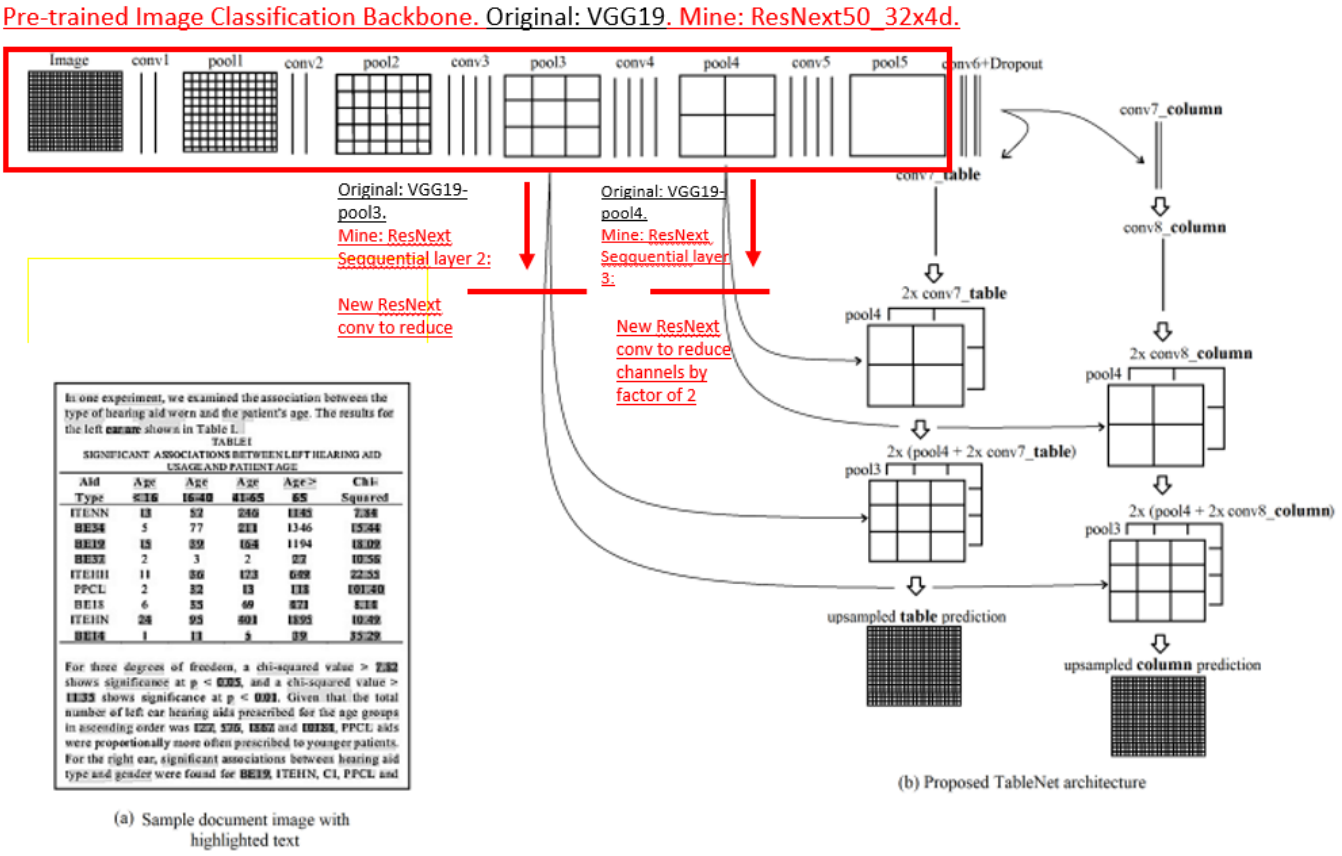

Figure 1. TableNet Original Annotated with Changes to ResNext Backbone

Experiment	Validation Table Accuracy	Validation Column Accuracy	Validation Table Precision	Validation Column Precision
TableNet – Original Baseline (VGG19) – 0.5 cutoff	0.970	0.957	0.910	0.828
TableNet with Batchnorm (VGG19) – 0.90 cutoff	0.971	0.962	0.900	0.832
TableNet with ResNext50_32x4d – 0.9 cutoff	0.941	0.934	0.877	0.785

Figure 2. Experimental Results on Validation Data

model specifications is monotonically decreasing. When unsmoothed, the original specification is the least noisy in its gradient descent, while the VGG19 with batch normalization specification is the noisiest.

Considering the selected metrics of interest, the replicated version of TableNet performed very similarly to the original findings from the paper for both training and validation datasets. One common trend was that table accuracy exceeded column accuracy, which is to be expected as columns are a special subcase of table detection. The original paper specification with and without batch normalization were relatively similar, achieving table validation accuracy of 97%, column accuracy of 96%, table precision of 90-91% and column precision lower in the 83% range. My new specification of TableNet with ResNext50_32x4d performed slightly worse across all metrics, only achiev-

Figure 3. Experimental Results - Loss Profile

ing 94% table accuracy, 93% column accuracy, 88% table precision, and 79% column precision. The monotonically increasing accuracy and precision metrics seem to confirm that the model is not overfitting to the training data (although it is possible that the model is overfit to the entire dataset, given that the validation data comes from the same source). However, one concerning trend in these measures is the poor performance of precision for the ResNext, particularly on the validation data. This seems to suggest that the ResNext model is making more predictions outside of the expected table and column mask regions and has an issue with erratic prediction. Qualitative results help further make this point.

The following figures show some examples of success and failure cases for the original TableNet with VGG19 and TableNet with ResNext versions. In both cases, the models are successful when document images are limited to having a small number of tables, with no other box-like figures such as graphs or diagrams. Conversely, TableNet fails in cases where there are numerous box-like shapes in the document, or if there is lots of text in the tables that look like non-tabular written sections of a document. Contrasting the VGG19 and ResNext versions of TableNet, it is clear that the original implementation is more conservative in its predictions, only predicting within a confined range. However, as shown by the ResNext failure case, in cases when there are more box-like objects, the model is more willing to predict erratically, even in small patches. This insight is matched by the quantitative precision findings, and suggests that the model is less certain about exactly what defines a table.

Saliency map analysis helps to show which features (pixels) of the input most influence the prediction in the output by visualizing the image gradient. Saliency maps of a successful TableNet with VGG19 example show that the model correctly identifies the table by finding the rough outline of the table, in particular its lower bounding line. It then focuses on the lines defining the columns to correctly split the table. However, the failure case shows that the model is thrown off by text in the middle of the document image, and fails to focus on a large inner section of the table. Similarly the TableNet with ResNext successfully identifies many of the key dividing white spaces around the document image to locate the table, but then hones in on tabular features like rows and columns in the bottom left corner.

5.4. Experiment Takeaways

Overall, these quantitative and qualitative results show that a relatively simple model in the original TableNet with VGG19, can be very successful in detecting both table and column masks. It is not clear why the ResNext version is less successful, but it is possible that just like the original ResNet's insight that the issue with deeper and wider networks is an optimization problem, not a design problem, may hold. That is, perhaps it would take much longer to get the ResNext backbone to generate high-fidelity lower-level features of use to the TableNet decoder. This is supported by the continued monotonic improvement in the accuracy and loss of the model. Another possibility is the fact that this architecture required inserting additional convolutional layers to reduce the number of channels. This dilutes the lower-level table features found by the backbone. An alternative might have been to adjust the decoder to match the output features of the ResNext backbone. This would be a good test for a future experiment, along with other techniques to help ResNext converge faster to better features to ensure at least comparable performance to the VGG backbone.

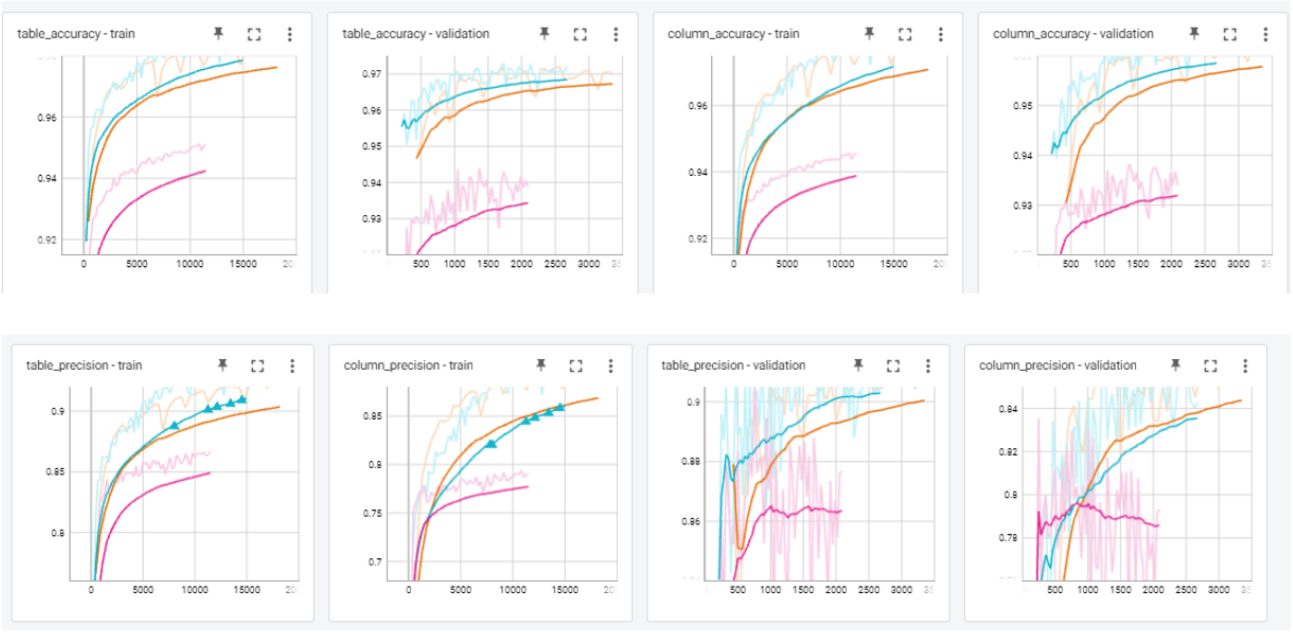

Figure 4. Experimental Results - Accuracy and Precision

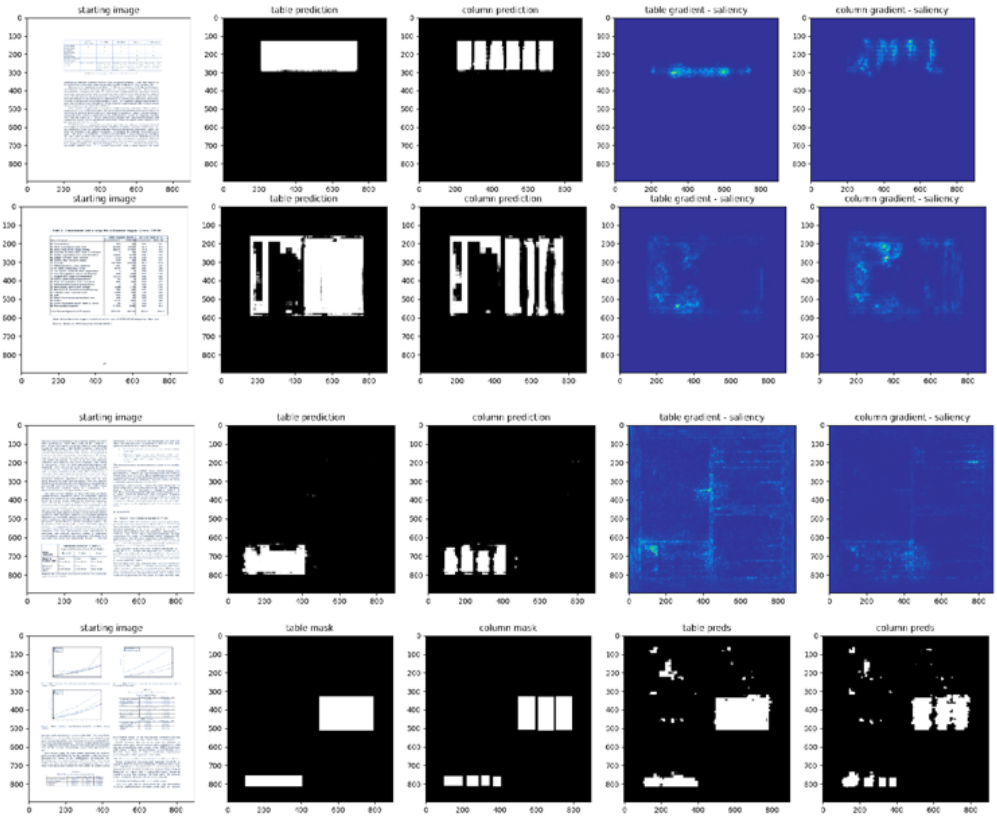

Figure 5. Experimental Results - Visualizations. Top: TableNet VGG19 Successful Predictions with Saliency Map, Second: TableNet VGG19 Failed Predictions with Saliency Map, Third: TableNet ResNext Successful Predictions with Saliency Map, Bottom: TableNet ResNext Failure Predictions with table and column masks shown

5.5. Conclusion/Future Work

In conclusion, utilizing an opensource implementation of TableNet, I have shown that TableNet is replicable and can be extended to use various alternative pre-trained backbone layers. While the ResNext backbone did not perform quite as well as the VGG19 layer, it is possible that future work could investigate this as an optimization problem, rather than solely as an architectural flaw. With additional time and resources, I would explore various alternative backbone architectures and encoder-decoder interfaces to test this claim.

Still, this new version did not perform poorly, achieving accuracy over 90%. In addition, visualization results have shown that all versions of TableNet focus on important parts of tables like boundaries and dividing lines, however the ResNext version also looks to the broader structure of the document, while the VGG19 version does a better job focusing only locally on the table. With additional time, it would have been interesting to further investigate learned features at various layer levels of the model, to identify the key activations being used to predict table and column masks. Another potential area of improvement would be to augment the dataset, particularly with documents that have more challenging combinations of both box-like figures and tables, as well as negative cases of table-less documents. This could serve both as a regularizer, as well as a way to help the model better optimize on learning features that uniquely pertain to tables.

The results of TableNet are promising as a semantic segmentation solution to the table detection and structure detection tasks, since very high performance is achievable with relatively simple architectures. Of course, as an end-to-end system, rules for converting from table masks to rows is required to make meaningful use of the predicted table and column masks. Each TableNet and the best performing instance segmentation method, CascadeTabNet, still must combine predictive and rules-based methods to fully extract meaningful table data from document images. Thus, one final meta-area of research in the future may be exploring the right balance between rules and prediction within the table detection and structure recognition task pipelines.

6. Acknowledgements

I would like to thank Tomas Sosorio for the open-sourced TableNet implementation upon which I built my testbench and model extensions. I would also like to thank the CS231n teaching staff for an excellent quarter.

References

[1] Madhav Agarwal, Ajoy Mondal, and C. V. Jawahar. Cdec-net: Composite deformable cascade network for table detection in document images. *CoRR*, abs/2008.10831, 2020. 2

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *CoRR*, abs/1906.07155, 2019. 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3

[4] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. Tablebank: Table benchmark for image-based table detection and recognition. *CoRR*, abs/1903.01949, 2019. 2

[5] Siwen Luo, Mengting Wu, Yiwen Gong, Wanying Zhou, and Josiah Poon. Deep structured feature networks for table detection and tabular data extraction from scanned financial document images. *CoRR*, abs/2102.10287, 2021. 2

[6] Shubham Paliwal, Vishwanath D, Rohit Rahul, Monika Sharma, and Lovekesh Vig. Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images. *CoRR*, abs/2001.01469, 2020. 1

[7] Devashish Prasad, Ayan Gadpal, Kshitij Kapadni, Manish Visave, and Kavita Sultanpure. Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents. *CoRR*, abs/2004.12629, 2020. 2

[8] Sachin Raja, Ajoy Mondal, and C. V. Jawahar. Table structure recognition using top-down and bottom-up cues. *CoRR*, abs/2010.04565, 2020. 2

[9] Christian Reisswig, Anoop R. Katti, Marco Spinaci, and Johannes Höhne. Chargrid-ocr: End-to-end trainable optical character recognition through semantic segmentation and object detection. *CoRR*, abs/1909.04469, 2019. 2

[10] Mausoom Sarkar, Milan Aggarwal, Arneh Jain, Hiresh Gupta, and Balaji Krishnamurthy. Document structure extraction for forms using very high resolution semantic segmentation. *CoRR*, abs/1911.12170, 2019. 1

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 1

[12] Tomas Sosorio. Ocr tablenet. *GitHub*, 2021. 1, 2

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 3

[14] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. 3