MLP Coursework 2: Exploring Convolutional Networks

s1874193

Abstract

This report explores convolutional networks with a focus on approaches to modelling context. More specifically, this report investigates the effects of striding and dilation on the EMNIST dataset. By adjusting the stride and dilation values context can be modelled differently, creating different hidden representations of the data. This report investigates the stride size, dilation size, the number of filters and the effect of regularisation in relation to their impact on model performance in an attempt to determine how to effectively model context on the EMNIST dataset. The results indicate that dilated convolutions are more capable of capturing and representing context within a character recognition setting if they are suitably regularised.

1. Introduction

Convolutional neural networks (CNN's) offer a highly effective approach to learning from images. First introduced by LeCun et al. (1998) and more recently popularised by Krizhevsky et al. (2012) in their famous ImageNet paper, CNN's have become the dominant approach to image-based machine learning tasks. This report investigates the development of CNN's and explores how different approaches of modelling context (striding and dilation) impact CNN performance. A CNN is implemented in *NumPy* by making use of the *SciPy* (Jones et al., 2014) module `scipy.signal.correlate2d`. Modelling context is then explored with a CNN developed in *PyTorch* (Paszke et al., 2017) and trained using a Nvidia K80 GPU on Google's Compute Engine. Later sections explore adjusting stride and dilation parameters to determine which approach is most effective on the EMNIST dataset (Cohen et al., 2017). The best values for each are then used to explore how the number of filters in each layer impacts performance for both striding and dilation. Finally, regularisation is explored on the two approaches.

The EMNIST dataset is an extension of the widely popular MNIST dataset (LeCun, 1998) consisting of handwritten digits. EMNIST extends MNIST by creating a dataset of both letters and digits. This extended dataset offers a more challenging and realistic recognition problem while sharing the same structure as the original dataset. Additionally, given the prevalence of the MNIST dataset, it is likely that most published works using MNSIT are overfitting to the literature to some extent. By offering a new, extended version, EMNIST reduces overfitting to the literature, providing an effective measure of performance. The dataset consists of 1.316×10^5 records, with 1×10^5 records in the training set and 1.58×10^4 records in both the testing and validation sets.

The remainder of this report will explore implementation and context modelling with CNN's. Section 2 describes the implementation of a CNN, Section 3 explores modelling context, Section 4 presents the experiments conducted, Section 5 offers a discussion on results and Section 6 concludes the work.

2. Implementing convolutional networks

Convolutional networks primarily consist of convolutional and pooling layers. These layers are an alternative approach to the affine layers of a feed-forward network that are able to capture and represent image data more accurately. Convolutional layers compute the dot product between a kernel (weights) and a small window of the input image. The window slides across the input image to produce an output. Typically, for a convolution operation, the kernel is flipped however for CNN's the kernel flipping is not necessary. In this case, the kernel is not flipped, which is equivalent to *correlation*

Pooling layers serve as a way of reducing dimensionality. Pooling slides over the input image in the same way as a convolutional layer, however, it does not have any learnable parameters. Instead, the pooling layer applies a simple operation, typically averaging or getting the maximum value for each window. By averaging or taking the maximum value, the pooling layer summarises the input layer into a more concise form.

For this report, a CNN was implemented in NumPy and SciPy by making use of SciPy's `correlate2d` function. The `correlate2d` function takes two, two-dimensional matrices as input, correlating one with the other. Typically the input to a convolutional layer is four-dimensional, so it was necessary to iterate through the input dimensions to obtain each two-dimensional image and kernel. Figure 1 provides some intuition for visualising the convolution operation.

The described approach offers an intuitive way of implementing convolutional layers as typically described in the literature; however, the number of loops needed makes this implementation very inefficient. An alternative approach would be to stretch out the four-dimensional input into a single matrix multiplication using the common `im2col` op-

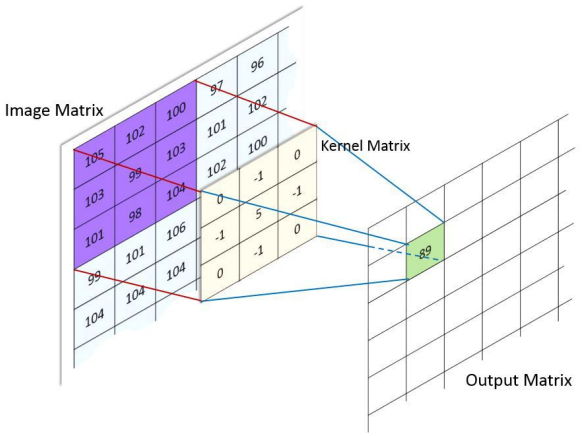


Figure 1. (Guru) The convolution operation being applied to a single image. The kernel matrix would then slide along the image to generate the next output value.

eration. By changing the convolution layers to a single matrix multiplication instead of multiple nested loops, we can make use of the highly optimised linear algebra implementations used in NumPy, specified by *Basic Linear Algebra Subprograms (BLAS)*. A naive BLAS implementation can perform matrix multiplication with a time complexity of $O(N^3)$, whereas the `correlate2d` implementation used for forward propagation has a time complexity $O(N^7)$. There is an additional overhead of running the `im2col` method itself, however, it only needs to be run once, storing the outputs in a cache. Consequently, the `im2col` approach is significantly more efficient, however, it is less intuitive to implement as it does not directly follow the standard convolution descriptions. While being more computationally efficient, `im2col` is notably less memory efficient as there is a large amount of data duplication. Despite its increase in memory overhead, `im2col` is typically used in practice due to its improved speed, and the additional memory overhead is rarely an issue in practice. Further increases in speed can be obtained through using *Fast Fourier Transforms* (Vasilache et al., 2014) and *Fast Algorithms* (Lavin & Gray, 2016), however they are beyond the scope of this report.

3. Context in convolutional networks

CNN's allow the representation of information at multiple scales, creating high-order abstractions of images that can be highly effective for computer vision tasks. Each representation builds upon previous representations to progressively represent the image in a more general context. Pooling, striding and dilation are all approaches to modelling this progressively increasing context.

As previously described, pooling layers summarise the input layer by applying a simple operation, typically averaging or taking the maximum value, although other approaches are also often used. The pooling layer replaces the output of the previous layer with a simple summary of each location and its neighbours. By representing each layer with simplified summaries, the representation becomes more *invariant* to small translations of the input, allowing the network to develop a more generalised representation of the task at hand instead of learning specific pixel locations of the features of an image (Goodfellow et al., 2016).

Instead of summarising locations on the input layer, striding summarises the input differently by specifying the amount by which the kernel moves when sliding across the input. By using a large stride value, the kernel is multiplied by the input less frequently, giving a more generalised context than a smaller stride value. Both striding and pooling reduce the dimensionality of the input, however, they result in different representations. While pooling is designed to improve translation invariance, striding is not, allowing positional information to be taken into greater account. However, as pooling is parameterless and striding is not, strided convolutions can take longer to train, although not as long as a standard convolutional layer.

A third approach to modelling context in CNN's is dilation. Dilation offers an approach to combining image representations at multiple scales without increasing the number of learnable parameters (Yu & Koltun, 2015). As shown in Figure 2, dilation expands the receptive field by increasing the distance between each filtered pixel. As in (Yu & Koltun, 2015) the size of the dilation is typically increased with each layer, creating an expanding receptive field and reducing dimensionality as the receptive field grows.

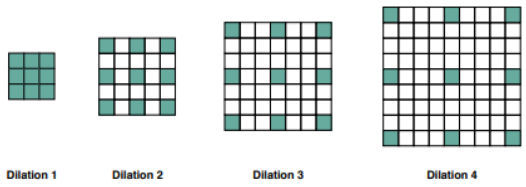


Figure 2. (Antoniou et al., 2018) The dilation operation at multiple scales. With each subsequent layer the dilation will increase from left to right, increasing the receptive field size and reducing the dimensionality of the output.

As an example, assuming a kernel w and input x, a standard convolution (dilation = 1) would be written as $w[0] * x[0] + w[1] * x[1] + w[2] * x[2]$. A dilation of 2 would be written as $w[0] * x[0] + w[1] * x[2] + w[2] * x[4]$. This gap between filtered pixels causes the receptive field to increase in size. As the gap widens with each layer the receptive filter continues to grow, creating a progressively widening context.

This report specifically addresses striding and dilated convolutions. Striding and dilation work in a similar manner; they both perform a dot-product between the input and kernel, and both adjust where the dot-product occurs, albeit in different ways. The purpose of the remainder of this report is to empirically and theoretically evaluate how these two approaches differ and determine which is most capable at modelling context for a character recognition task

Despite there being many possible research questions regarding pooling, it has not been investigated in this report.

Due to similar approaches used for both striding and dilation, they immediately present themselves for comparison. Additionally, Springenberg et al. (2014) find that max-pooling can be replaced by striding in many cases, meaning conclusions reached regarding striding convolutions, may be generalised to some extent to max-pooling.

4. Experiments

For all experiments discussed the same default architecture was used. This architecture consisted of 4 layers, where each layer consists of a convolution, followed by a ReLU layer, a dilation or striding layer, and a second ReLU layer. After these four layers, there is a final linear layer and an output layer. The optimisation method used was Adam with weight decay (Loshchilov & Hutter, 2017), with weight decay set to 1×10^{-5}. A batch size of 100 and a random seed of 0 was used for every experiment to ensure all the experiments are reproducible.

This model was then used to run two baselines, comparing striding and dilation. The default stride size was set to two, and the dilation rate was set to $l + 2$, where l is the current layer of the network. As a starting point, the number of filters was set to 32. Results of the baselines on the test data can be seen in Table 1. All subsequent experiments will be performed on the validation data.

	Strided Convolution	Dilated Convolution
Accuracy	86.9937%	88.0316%

Table 1. Baseline accuracy percentages of both striding and dilation on the test set

As the above results show, dilated convolutions perform best with the baseline parameter settings. To fully understand the differences and similarities between dilation and striding further experiments should be run that explore how each approach performs with different hyperparameters. By doing a thorough evaluation of each approach in multiple settings we can gain a greater understanding of how striding and dilation compare to each other. The experiments begin by finding desirable striding and dilation values. Determining the best hyperparameter settings of each approach allows a fairer comparison between the two by ensuring both models are performing optimally. We then investigate how both striding and dilation performance varies with the number of filters and how this impacts overfitting.

4.1. Varying Stride and Dilation Values

The stride and dilation values were set arbitrarily for the baseline experiments; by adjusting these values the best value for each can be found, and more accurate comparison between the two can be made. By default, dilation increases with the number of layers, whereas striding is constant. Experiments were conducted that varied the stride and dilation sizes, as well as trying constant and increasing dilation sizes (Table 2).

Reduction Type	Value	Accuracy
Stride	1	89.4430%
	2	88.2025%
	3	87.5253%
	4	86.5217%
	5	2.0759%
Dilation	1	89.4241%
	2	89.3354%
	3	88.8797%
	4	88.5569%
	$l + 1$	89.1202%
	$l + 2$	89.0063%

Table 2. Performance on the validation set of striding and dilated convolutions as their respective value varies. A dilation value of 1 is equivalent to no dilation.

Varying the stride value produced some interesting results. A stride value of one gave the best performance, and as the stride is increased, performance gradually drops off. Once stride reaches five, performance drops dramatically. A value of five is too large and unable to capture anything meaningful from the data.
With a constant dilation value, the best performance was achieved with a value of one. As the dilation is changed with the current layer, the best performance was achieved with dilation set to $l + 1$, although it did not perform as well as the best constant value. While a constant value of one gives the best performance no dilation is being performed; this is the same dimensionality reduction as a striding convolution with a value of 1, resulting in very similar performance. Consequently, a constant dilation value of one is not useful for making a comparison between the two approaches and can be dismissed, making the best performing dilation value $l + 1$. The stride and dilation values were then set to the best values for all future experiments.

4.2. Varying the Number of Filters

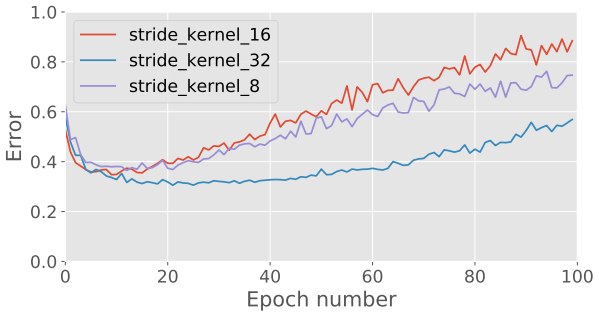
Having determined the best stride and dilation values the next point of interest is the number of filters. By changing the number of filters the CNN's ability to capture and represent features of an image is changed. As the number of filters increases the representational power of the model is increased. However, as representational power increases so does the model's capacity to overfit. Selecting a suitable number of filters is, therefore, of importance when trying to improve a model's performance and is of interest when comparing both striding and dilated convolutions. Does one approach require more filters than the other to learn adequately? Does either approach overfit, and by adjusting the number of filters can that be resolved? By varying the number of filters, these questions can be explored.

Table 3 shows the results after training the model with different numbers of filters. Interestingly, striding performs best with 32 filters, whereas dilation performs best with 16

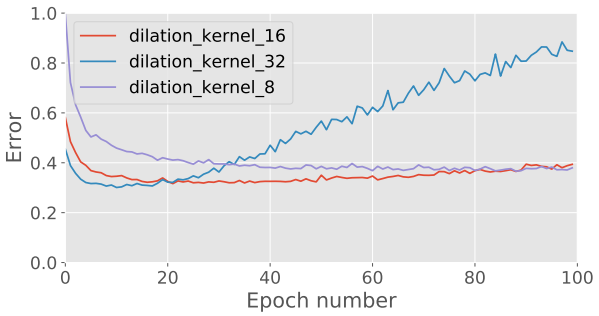
	STRIDING	DILATION
8 FILTERS	87.9937%	89.0126%
16 FILTERS	87.9810%	89.1962%
32 FILTERS	89.3101%	87.4620%

Table 3. Striding and dilation performance as the number of filters is varied. The results suggest that dilation begins to overfit with 32 filters, whereas striding does not.

filters. As the number of filters increases the striding performance also increases, whereas dilation begins to decrease again after 16 filters. This decrease in dilation performance with 32 filters suggests that the model is overfitting. By adding more filters, there are more learnable parameters, making the model more likely to overfit. Additionally, it appears that striding performance is poor with both 8 and 16 filters, requiring 32 filters to perform adequately. In contrast, dilation performs well with both 8 and 16 filters, and only loses performance with 32 filters. Figure 3 shows a plot of the validation error over time, confirming that dilation is overfitting with 32 filters. Figure 3 additionally shows that the striding validation error begins to increase again as training progresses, suggesting that it is also overfitting. By resolving the overfitting problem, it may be possible to improve results for both approaches further.



(a)



(b)

Figure 3. Error as the number of filters is adjusted for stride (a) and dilation (b)

4.3. Regularisation

Dropout is a form of regularisation designed for deep neural nets (Srivastava et al., 2014), which has been successfully applied to CNN's (Wu & Gu, 2015). Dropout works by dropping a proportion of the hidden units in a network by applying a binary mask; this reduces the number of parameters and minimises overfitting. As striding performs best with 32 filters and performance is increasing with each increase in the number of filters dropout will be applied to this model to improve performance further. For dilation, the performance stops improving after 16 filters due to overfitting. By applying dropout to the dilation model with 32 filters, it may be possible to resolve the issue of overfitting and improve generalisation error due to the increased representational power of the additional filters.

Following Park & Kwak (2016), dropout was applied after the final non-linearity in each layer of the network with a dropout probability of $p = 0.2$. The results of applying dropout to striding and dilation can be seen in Table 4.

	STRIDING	DILATION
NO DROPOUT	89.3101%	87.4620%
DROPOUT	89.4747%	89.9999%

Table 4. Dropout performance for striding and dilation, both with 32 filters. Dilation performance has significantly improved, striding has had a slight improvement.

The results show that the addition of dropout has improved results for both striding and dilation. Of particular interest is the increase in performance for the dilation model. Dilation performance increased significantly more than striding, making it now the best performing approach. Comparing the results between Table 3 and Table 4, it is clear that by regularising the dilation model it is no longer overfitting, allowing the model to make use of the additional representational power provided by increasing the number of filters without learning the training data too closely. The performance of the striding model has also increased, however not to the same extent. As the striding model with 32 filters was overfitting less extensively a smaller increase in accuracy is expected. Figure 4 shows the validation curves of both models. A comparison between Figure 3 and Figure 4 highlights the impact of dropout. Both models are no longer overfitting at all, with dropout slightly outperforming striding.

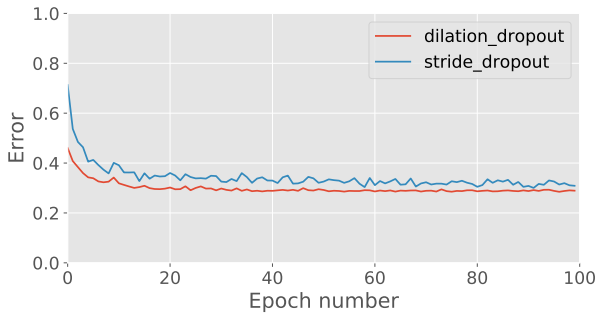


Figure 4. Validation error over time after applying dropout to both the dilation and stride models.

After applying dropout, both approaches achieved their best results. The final models for each were then evaluated on the test set. Results with a standard error can be seen in Table 5.

	STRIDING	DILATION
ACCURACY	88.6350% ± 0.00049	89.0612% ± 0.00035

Table 5. Final performance with error bars on the test for both striding and dilation

The standard error was calculated as shown below (1).

$$\frac{standard\ deviation}{\sqrt{number\ of\ experiments}} \tag{1}$$

The results indicate the outcome of each approach, showing that dilation will consistently perform better and that both striding and dilation can be significantly improved given suitable regularisation and hyperparameter choices.

5. Discussion

The experiments conducted in the previous chapter provide insight into striding and dilation and their ability to perform on the EMNIST dataset. The initial baseline showed that with little thought to hyperparameters, dilation performs better than striding. Van Den Oord et al. (2016) offer some intuition behind dilation's good performance, stating that dilation allows a large receptive field over a small number of layers, allowing high-order abstractions to be represented by the network. After tuning the hyperparameters a more accurate representation of each approaches performance is obtained. With a value of 1, striding can slightly outperform dilation, which does not support Van Den Oord et al. (2016)'s hypothesis. Constant dilation values and values that scale linearly with the current layer are tested. These approaches are effective although are not as effective as a well-tuned striding network. Some literature (Yu & Koltun, 2015) suggests using an exponentially increasing dilation rate, which may lead to better performance.

After tuning the striding and dilation values, both approaches were evaluated with varying numbers of filters. By adjusting the number of filters the representational power of the network changes. More filters gives more parameters that can be tuned, although there is more potential to overfit. By comparing performance with varying numbers of filters the ability of each approach to model context can be evaluated. The results indicated that dilation was able to outperform striding with fewer filters, although it began to overfit with 32 filters. The comparatively better performance of dilation with fewer filters suggests that it is more capable of capturing and representing information from within the image, supporting the intuition posited by Yu & Koltun (2015) and Van Den Oord et al. (2016). As the number of filters increases dilation begins to overfit while striding continues to improve. These results support the idea that dilation offers increased context in a small number of layers; with 32 filters dilation is very capable of closely representing the input data to the point of overfitting, whereas striding is less capable of capturing representations and, therefore, requires more filters to produce a good result.

Finally, dropout was implemented to combat overfitting. With suitable regularisation, dilation should be able to continue to improve with the number of filters, as its representational power can increase without overfitting. Striding performance should also improve although it is overfitting less so the performance increase will not be as significant. The results support these conclusions, showing that with effective regularisation, dilation can outperform striding on the test data. For the final test evaluations, error bars were calculated to ensure correct conclusions could be reached. On average, dilated convolutions outperformed striding convolutions when both approaches were suitably optimised and regularised. The error bars of striding and dilation did not overlap, indicating that the results will be consistent for future evaluations.

6. Conclusions

To conclude, the question posed at the beginning of this report was to compare striding and dilated convolutions to examine which approach was most effective at modelling context for a simple character recognition task. Through a serious of experiments, this report showed that dilation was more capable at modelling context due to its ability to rapidly expand its receptive field, generating high-order representations of an image in a small number of layers. The experiments began by tuning each model, and then evaluating how each performed with varying numbers of filters. Striding performance continued to improve as the number of filters increased, whereas dilation began to overfit with 32 filters. This overfitting lead to the implementation of dropout and further experimentation that suggested how dilation is more capable of capturing context than striding.

6.1. Further work

Having established that dilation can outperform striding on EMNIST, it would be of interest to explore how striding and dilation compare on more complex tasks. The quickly expanding receptive field of dilation should allow it to capture greater levels of invariance and correlations between sections of images that are spread out. A dataset such as CIFAR-10 (Krizhevsky & Hinton, 2009) would allow for further empirical testing of dilated convolutions increased ability to model context.

Additional work could also explore further techniques of adjusting the dilation with the number of layers. As mentioned in the discussion, some literature suggests using an exponentially increasing dilation to increase the receptive field more rapidly. Similar approaches may also be tested for striding, however, the results in Table 2 show that with a stride value of 5 performance is dramatically reduced. Con-

sequently, an increasing stride value would quickly become too large.

References

Antoniou, Antreas, Słowik, Agnieszka, Crowley, Elliot J, and Storkey, Amos. Dilated densenets for relational reasoning. *arXiv preprint arXiv:1811.00410*, 2018.

Cohen, Gregory, Afshar, Saeed, Tapson, Jonathan, and van Schaik, André. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

Goodfellow, Ian, Bengio, Yoshua, Courville, Aaron, and Bengio, Yoshua. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Guru, Machine Learning. Imagefiltering-imageconvolution. URL http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html. Accessed: 20-11-2018.

Jones, Eric, Oliphant, Travis, and Peterson, Pearu. {SciPy}: open source scientific tools for {Python}. 2014.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lavin, Andrew and Gray, Scott. Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4013–4021, 2016.

LeCun, Yann. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Loshchilov, Ilya and Hutter, Frank. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.

Park, Sungheon and Kwak, Nojun. Analysis on the dropout effect in convolutional neural networks. In *Asian Conference on Computer Vision*, pp. 189–204. Springer, 2016.

Paszke, Adam, Gross, Sam, Chintala, Soumith, Chanan, Gregory, Yang, Edward, DeVito, Zachary, Lin, Zeming, Desmaison, Alban, Antiga, Luca, and Lerer, Adam. Automatic differentiation in pytorch. 2017.

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Van Den Oord, Aäron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew W, and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. In *SSW*, pp. 125, 2016.

Vasilache, Nicolas, Johnson, Jeff, Mathieu, Michael, Chintala, Soumith, Piantino, Serkan, and LeCun, Yann. Fast convolutional nets with fbfft: A gpu performance evaluation. *arXiv preprint arXiv:1412.7580*, 2014.

Wu, Haibing and Gu, Xiaodong. Towards dropout training for convolutional neural networks. *Neural Networks*, 71: 1–10, 2015.

Yu, Fisher and Koltun, Vladlen. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.