

Data Generation with Convolutional Neural Networks

Marvin Krauß

August 18, 2025

Contents

1	Introduction	2
2	What are Convolutions	2
2.1	Transposed Convolution	2
2.2	Convolutional Neural Networks	2
2.3	Receptive Field	2
3	Generative Model Approaches	3
4	Autoregressive Image Generation with PixelCNN	3
4.1	Blind Spot	3
4.2	Gated Blocks	4
4.3	Training	5
4.4	Sampling	5
4.5	Conditional PixelCNN	5
4.6	VAE with conditional PixelCNN	5
4.7	Discretized Logistic Mixture Likelihood	6
4.8	Experiments	6
5	Sound Generation with WaveNet	6
5.1	Shifted Convolutions	7
5.2	Gating Block and Conditioning	7
5.3	Exceptional Input Length	7
5.4	Experiments	8
6	Text Generation	8
7	Conclusion	8

1 Introduction

Convolutional Neural Networks have already been used for data classification with great success and have shown huge potential. In this paper I will show how CNNs can be used to not classify data, but generate it. Precisely i will show its generation capabilities for images, audio and text. Additional tests and empiric data will show, that CNNs have potential in generating data.

2 What are Convolutions

A convolution slides a small filter, also called kernel, over the input data, computing the weighted sum of its positional corresponding input elements. The resulting matrix is called feature map. The weights are the values of the filter. Since the filter does not fit over edge elements this results in a decrease of spatial size. For the resulting spatial size to stay the same, zero valued elements called padding are added at the edges of the input, so that the filter fits on every element. Dilation refers gaps between the filter elements increasing filter range without increasing parameter size.

$$Output_Size = Input_Size - \left\lfloor \frac{Kernel_Size}{2} \right\rfloor * Dilation + Padding$$

2.1 Transposed Convolution

Default convolutions decrease spatial size, transposed convolutions increase spatial size. This time, each element is multiplied by the filter and put the resulting matrix into the feature map. The distance between each matrix is called stride. Overlapping elements in the feature map are summed. Here, padding removes edge elements of the feature map.

$$Output_Size = (Input_Size - 1) * Stride + Kernel_Size - 2 * Padding$$

2.2 Convolutional Neural Networks

Convolutional Neural Networks work by stacking multiple convolutions with each multiple feature maps. The values of the filters are the learnable weights adjusted by backpropagation. The network then learns to extract local structural information. This information can then be used for further processing like classifiers.

2.3 Receptive Field

The receptive field of an element is the reach in which its neighbors can influence the element. A larger receptive field means the possibility to capture larger structural information. This is achieved by either increasing filter size, dilation or using multiple

convolutions one after another. While the other two options increase parameter size of the network, which will result in higher complexity and computation costs, dilation will not. On the other side using too much dilation can lead to neglect of local structures. When a convolution is used elements will gather information from their neighbors meaning that after the next convolution elements receive information indirectly from their neighbors' neighbors.

3 Generative Model Approaches

There are different generative model concepts like Variational Auto Encoders that use a CNN for encoding and a transposed CNN for decoding or Generative Adversarial Models that use a transposed CNN for creating data and a CNN for categorizing the generated data into real and fake. In the following text I will focus on the autoregressive approach and with that at first on images. Specifically we will look at the PixelCNN architecture.

4 Autoregressive Image Generation with PixelCNN

In the autoregressive image generation each pixels probability is based on its predecessors. The predecessors are relative to the pixel being computed positioned on the rows above or on the same row on the left. $p(\mathbf{x}) = \prod_{i=1}^N p(x_i | x_1, x_2, \dots, x_{i-1})$. This is achieved by masking the filters by assigning zero value to the successor weights. On RGB images this masking is also applied to the color channels meaning that on the current pixel red channel can only see previous pixels. The green channel can only see previous pixels and red channel and the blue channel can see previous pixels, red and green channel. This mask is only applied on the first layer whereas on the remaining layers a mask is used where the current channel can see itself too. This works because the first layer filters out the unwanted information of the current channel resulting in the next layers receiving only generated information about current channel.

4.1 Blind Spot

With the current masking a blind spot occurs that blacks out up to nearly one half of the receptive field. This happens because each pixel receives information further away indirectly from its neighbors, but its neighbors do not look to their right since those are their successors. This leads to that pixels cannot forward all necessary information downwards. This can be avoided by implementing two separate stacks, the horizontal stack looking at all pixels on the same row to the left and the vertical stack looking at all pixels above. The information of the vertical stack is then feed to the horizontal stack. The vertical stack will grow to each horizontal side and upwards and start at the pixel

above the current center. That ensures the pixels in the blind spot will be included as well and since the vertical stack starts from the row above it is allowed to look to the right.

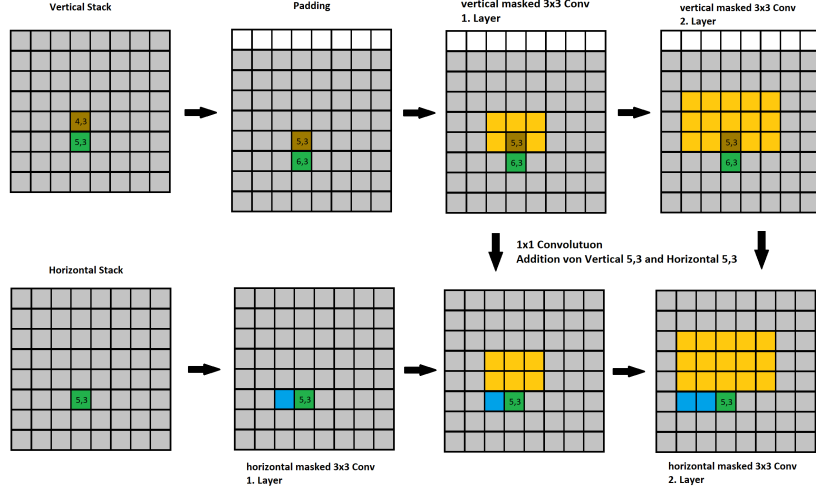


Figure 1: Implementation of the horizontal and vertical stack with a 3x3 kernel

4.2 Gated Blocks

In the original Model a conventional non linear activation function like the rectified linear unit, but with inspiration from LSTM Models a gating mechanism was introduced. Now each layer produces two times the feature maps where one half goes into the tanh function and the other half into the sigmoid function. The tanh function produces the information or signal in value range $\{-1, 1\}$ and the sigmoid function produces the gating value ranging $\{0, 1\}$. The gating value decides, how much information is passed through. Furthermore the horizontal stack utilizes a residual connection since those help deep networks train faster and help with vanishing gradients.

$$Y_{vert} = \sigma(W_{vert, gate} * x_{vert}) \circ \tanh(W_{vert, signal} * x_{vert})$$

$$Y_{hor} = x_{hor} + W_{out} * \left(\sigma(W_{hor, gate} * x_{hor} + W_{conv} * x_{vert}) \right) \circ \tanh(W_{hor, signal} * x_{hor} + W_{conv} * x_{vert})$$

Here $*$ is the convolution operator, W the corresponding convolution, w_{conv} is the convolution for converting the vertical stack for the horizontal stack and $\circ$ the element-wise product.

4.3 Training

The input for the model are the training data images. Since all predecessors are already known all pixel can be computed parallel. The default model predicts the probability distribution for the color values for each channel for each pixel by using softmax. The loss is the negative log likelihood.

4.4 Sampling

Since each pixel is based on its predecessors, the model predicts each pixel sequentially. For example a 64 x 64 RGB image needs 12.288 runs until finished. This results in huge sampling times. Since successor elements are not needed for calculation, the input matrix is cut from bottom to the to be calculated pixel for reducing sampling time.

4.5 Conditional PixelCNN

In a conditional PixelCNN we have a latent vector $\mathbf{h}$ on which all pixels are based on $p(\mathbf{x}) = \prod_{i=1}^N p(x_i | x_1, x_2, \dots, x_{i-1}, \mathbf{h})$. If $\mathbf{h}$ does only contain location independent information, meaning information about what to show not where, a linear projection of $\mathbf{h}$ is broadcast across spatial dimensions on each layer and stack.

If $\mathbf{h}$ is location dependent a transposed CNN is used to bring $\mathbf{h}$ to the dimensions of $\mathbf{x}$. Next a 1x1 convolution is used to bring the arbitrary amount of feature maps of the transposed CNN to the amount of feature maps of $\mathbf{x}$. The new gated block with the bias looks like this:

$$Y_{vert} = \sigma\left(W_{\text{vert, gate}} * (x_{vert} + bias)\right) \circ \tanh\left(W_{\text{vert, signal}} * (x_{vert} + bias)\right)$$

$$Y_{hor} = x_{hor} + W_{out} * \left(\sigma\left(W_{hor, gate} * (x_{hor} + bias) + W_{conv} * (x_{vert} + bias)\right) \right. \\ \left. \circ \tanh\left(W_{hor, signal} * (x_{hor} + bias) + W_{conv} * (x_{vert} + bias)\right) \right)$$

4.6 VAE with conditional PixelCNN

For improving the quality and sampling time a conditional PixelCNN can be used as decoder in a Variational Auto Encoder. The PixelCNN will be conditioned on the latent vector produced by the encoder. The advantage of this approach is that the encoder will capture global structure information fast and feed it to the PixelCNN which then only has to focus on local structures. This leads to the PixelCNN needing a smaller receptive field and thus less layers and parameters. Since that is the sampling bottleneck sampling

time will be improved.
experiments

4.7 Discretized Logistic Mixture Likelihood

Another improvement is the change from categorical softmax distribution to Discretized Logistic Mixture Likelihood. The model now computes the the mean, scale and weight for a chosen amount of logistics functions. Since the color values are discrete values each color value is assigned the area of $\{y - 0.5, y + 0.5\}$. The edge values 0 and 255 are assigned the area $\{-\infty, y + 0.5\}$ and $\{y - 0.5, +\infty\}$. For training the negative log of the weighted sum of the probabilities of the logistics functions of the correct color value will be minimized. For sampling per softmax and multinomial one logistics function is chosen and with that the color value is calculated. One big advantage is that similar values like 234 and 233 are assigned similar probabilities which is reasonable. The softmax approach can and will do that too, but needs more training runs to learn this correlation.

4.8 Experiments

In the tables Figure 2 and Figure 3 the quality of the different approaches is compared.

Model	NLL Test (Train)	Model	Bits per sub-pixel
Uniform Distribution: [30]	8.00	Deep Diffusion (Sohl-Dickstein et al., 2015)	5.40
Multivariate Gaussian: [30]	4.70	NICE (Dinh et al., 2014)	4.48
NICE: [4]	4.48	DRAW (Gregor et al., 2015)	4.13
Deep Diffusion: [24]	4.20	Deep GMMs (van den Oord & Dambre, 2015)	4.00
DRAW: [9]	4.13	Conv DRAW (Gregor et al., 2016)	3.58
Deep GMMs: [31, 29]	4.00	Real NVP (Dinh et al., 2016)	3.49
Conv DRAW: [8]	3.58 (3.57)	PixelCNN (van den Oord et al., 2016b)	3.14
RIDE: [26, 30]	3.47	VAE with IAF (Kingma et al., 2016)	3.11
PixelCNN: [30]	3.14 (3.08)	Gated PixelCNN (van den Oord et al., 2016c)	3.03
PixelRNN: [30]	3.00 (2.93)	PixelRNN (van den Oord et al., 2016b)	3.00
Gated PixelCNN:	3.03 (2.90)	PixelCNN++	2.92

Figure 2: CIFAR-10, Bits/dim error, left: PixelCNN [4], right: PixelCNN++ [2]

5 Sound Generation with WaveNet

Similar to how PixelCNN captures local structures in images, WaveNet captures local structures in sound waves. The difference is that the input and thus convolutions are one-dimensional instead of 3-dimensional. The input data is raw audio waveforms, meaning time x air pressure.

Model	NLL Test
DRAW (Gregor et al., 2016)	≤ 80.97
Discrete VAE (Rolfe, 2016)	≈ 81.01
IAF VAE (Kingma et al., 2016)	≈ 79.88
PixelCNN (van den Oord et al., 2016a)	≈ 81.30
PixelRNN (van den Oord et al., 2016a)	≈ 79.20
Convolutional VAE	≤ 87.41
PixelVAE	≤ 80.64
Gated PixelCNN (our implementation)	≈ 80.10
Gated PixelVAE	$\approx 79.48 (\leq 80.02)$
Gated PixelVAE without upsampling	$\approx \mathbf{79.02} (\leq 79.66)$

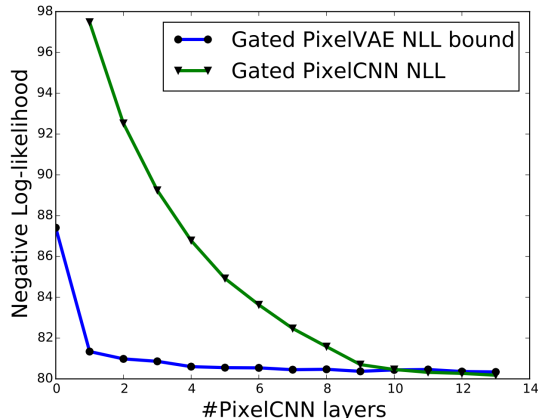


Figure 3: MNIST, PixelVAE [1], right: PixelVAE test results, left: Comparison layer effectiveness PixelCNN, PixelVAE

5.1 Shifted Convolutions

As opposed to masking, shifting can be used too to achieve causality. For mask b that allows the current element to see itself, the input data is added padding of size $\left\lfloor \frac{Kernel\ Size}{2} \right\rfloor * Dilation * 2$ for odd kernel size. For mask a, which does not let the current element see itself, the padding size is increased by one and the input datas' last row is dropped. Shifting is generally more complex in higher dimensions.

5.2 Gating Block and Conditioning

WaveNets' architecture is very similar to that of PixelCNN meaning both use gated blocks, though WaveNet does not use different stacks. Furthermore the conditioning process is the exact same with transposed CNNs for time dependent information and a simple linear projection for time independent information.

5.3 Exceptional Input Length

One second of audio with 16 kHz needs 16000 elements. This leads to the model needing a large receptive field. There are two major strategies for dealing with that, one is to use dilated blocks where the dilation for the next layer grows exponential cycling in blocks leading to dilation patterns like this $\{1, 2, 4, 8, \dots, 512, 1, 2, 4, 8, \dots, 512, 1, 2, 4, 8, \dots, 512\}$. The second strategy involves smaller models that have a large receptive field with few parameters called context stacks that feed their results as condition into a large model that has a smaller receptive field with more parameters. The context stacks focus on capturing global structural information with small complexity whereas the main model focuses on only local structure information with high complexity. Together these two strategies lead to the model being capable to capture the whole data without the complexity taking over.

5.4 Experiments

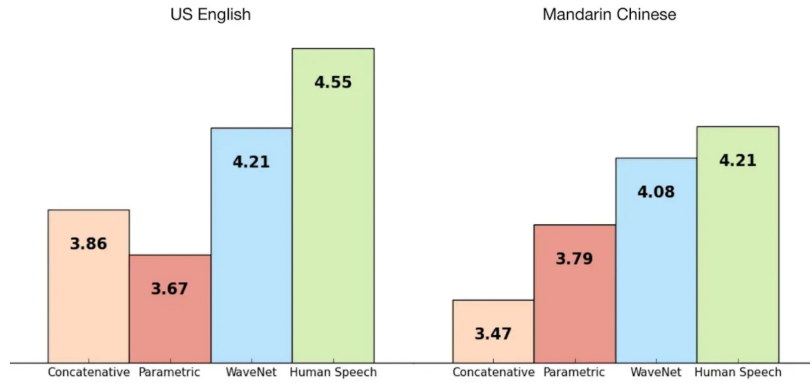


Figure 4: Comparison with MOS (Mean Opinion Score) between WaveNet, Concatenative (Segment stitching) und Parametric (LSTM-RNN) [3]

6 Text Generation

CNNs can also be used for text generation, where a sentence is tokenized and processed similar to WaveNet by one dimensional convolutions. An implementation of this approach is the genCNN model, which results are shown in the table below.

Model	Perplexity	Dynamic
5-gram, KN5	141.2	—
FFNN-LM	140.2	—
RNN	124.7	123.2
LSTM	126	117
<i>gen</i> CNN	116.4	106.3

Figure 5: PENN TREEBANK, [5]

7 Conclusion

In their time autoregressive CNN generators were state of the art and achieved many breakthroughs and records. Now other models like transformer based or diffusion based models have drastically passed the stated models in all categories. Non the less the findings and milestones of these architectures were groundbreaking and are partially found in newer state of the art models.

References

- [1] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images, 2017.
- [2] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017.
- [3] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.
- [4] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016.
- [5] Mingxuan Wang, Zhengdong Lu, Hang Li, Wenbin Jiang, and Qun Liu. gencnn: A convolutional architecture for word sequence prediction, 2015.