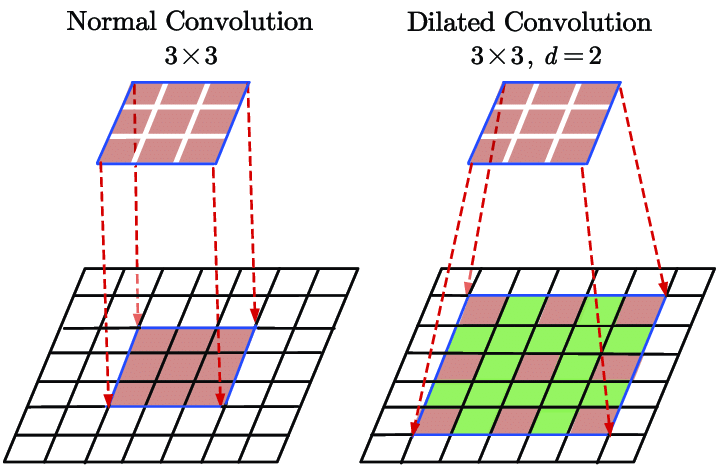
TCN Model Report

# General Description

Temporal Convolutional Networks (TCNs), as developed by Bai, Kolter, and Koltun, work to solve the main problem with traditional convolutional neural networks: small receptive field, or the inability to see much of the data at once. TCNs get past this problem by using dilation, as pictured below:



This allows each layer to skip some timestamps, meaning that the receptive field can grow exponentially with the size of the network, similar to a UNet.

The network consists of a stack of these convolutions, with the dilation increasing each time by some dilation base so the i-th layer will have a dilation of *dialation\_base*^i (traditionally base 2). Each subsequent layer incorporates information from a wider and wider view of the data. There are also skip connections between layers to pass the information from the low-level view of the data to higher-level views.

The wide receptive field and ability to incorporate information from a wide range of data makes this a good choice for our task. Traditionally, TCNs only use past information to predict the future which makes sense in contexts like finance (called causal convolutions). However, in our case, looking at the future data is not only acceptable but required for a full picture. What makes a TCN unique from a fully convolutional network (FCN) developed by Long et al is this causal convolution, making our TCN more of an FCN. For this reason, we are considering changing to the name of this model to FCN.

# Model

The architecture, at a high level, is:

* *num\_levels* repetitions of ResidualBlock, with the i-th block having a dilation of dilation\_base ^i. The in\_channels and out\_channels are decided by the list *num\_channels*, except the first layer takes *num\_inputs* and the last outputs *num\_classes*.
* Average pool over the input
* Linear layer with *num\_classes* (6) output

There are skip connections between the layers.

# References

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.” *arXiv*, 2018, [arXiv:1803.01271](https://arxiv.org/abs/1803.01271).

Long, Jonathan, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. *arXiv*,<https://arxiv.org/abs/1411.4038>.

Akiba, Takuya, et al. "Optuna: A Next-generation Hyperparameter Optimization Framework." *arXiv* (2019), arXiv:1907.10902.<https://arxiv.org/abs/1907.10902>.

# Technical Notes (for appendix)

The data is chunked into sections for prediction, with a single label for each. The model predicts a single label for a 0.5 second window, with 3 seconds before and 3 seconds after passed as additional context. The predictions are made on non-overlapping 0.5 second windows. The target label is given by the label in the middle of the window.

## Architecture

class TCN(nn.Module):

def \_\_init\_\_(self, num\_inputs, num\_channels, num\_classes, kernel\_size=2, dropout=0.2, dilation\_base=2)

The architecture, at a high level, is:

* *num\_levels* repetitions of ResidualBlock, with the i-th block having a dilation of dilation\_base ^i. The in\_channels and out\_channels are decided by the list *num\_channels*, except the first layer takes *num\_inputs* and the last outputs *num\_classes*.
* Average pool over the input
* Linear layer with *num\_classes* (6) output

## Residual Convolutional Block

class ResidualBlock(nn.Module):

def \_\_init\_\_(self, in\_channels, out\_channels, kernel\_size, dilation, dropout, is\_last=False)

Each residual block consists of 2x: conv, relu, dropout. If *in\_channels* != *out\_channels*, then a conv from *in\_channels* to *out\_channels* is run, followed by a relu. There is a skip connection from before the residual block to the end.

## Training

The model is trained with Adam with CrossEntropy loss, using the specified *lr* and *weight\_decay* and run for *epochs*.

As with all other models, the data is first split into probes.