[CS395] Final Project: Window-Based Activation Functions

Kelsey Ball
kelseyball@utexas.edu

Marco Bueso
mbueso@utexas.edu

Zayne Sprague
zaynesprague@utexas.edu

1. Introduction

From its origins, deep learning has drawn analogies from the mammalian brain, particularly finding inspiration on the human cerebral cortex [7]. One important analogy is how neurons transmit information. Biologically, neurons are complex signaling cells, which transmit electric pulses through "synapses" between adjacent cells. An electric pulse will only be forwarded by a neuron if the voltage on the receiving (post-synaptic) neuron reaches a threshold voltage, called "action potential". Neurons are interconnected, and multiple neural paths can converge into a single post-synaptic neuron's action potential. Scientists have modeled this neural activity for decades. Back in 1943, McCulloch and Pitts presented a simplified neuron model, based on the "all-or-none" activation on a neuron (the action potential), through propositional logic [5]. Using this model, perceptrons and multi-layer perceptrons followed, allowing for the first deep learning models.
Activation functions have been used in these networks since then, and are still a key component. Certain activation functions have provided really good results, with simple implementations. Perhaps the most common and simplest one is ReLU, the Rectified Linear Unit activation function. Due to its effectiveness, a lot of activation functions have been modeled after ReLU, essentially becoming fine-tuned variations of it (LeakyReLU, GELU, etc.). A lot of current research in activation functions focuses on iteratively improving ReLU via small adjustments.
However, we believe it's a good idea to branch off from the element-wise implementation of activation functions; drawing inspiration from the mammalian brain, we present a set of window-based activation function, which aim to resemble some of the biological neural network interactions. Particularly, we're interested in the synaptic inhibition phenomenon, which can be global or selective. Global synaptic inhibition occurs when a presynaptic neuron lowers the incoming signal to the post-synaptic neuron below the action potential threshold, resulting in no information passing to the downstream neurons or target cells. In selective presynaptic inhibition, an inhibitory neuron synapses on one branch of the transmitting neuron, and this inhibits only one of the multiple targets of the neuron [8]. Following the selective inhibition idea, we can formulate a system where the activation of a node in a neural network can selectively inhibit the transmission of information to neighboring nodes.
Parting from this idea, we explore different alternatives where a node's action potential can have a weighted effect on neighboring node activations, whether excitatory or inhibitory.

2. Related works

2.1. Local Normalizations

Similar functions have been implemented as normalization layers. Jarrett et. al [1] implement "local contrast normalization" also inspired by computational neuroscience. This module subtracts a Gaussian-weighted average of a local neighborhood from each value in a feature map, then divides by a weighted standard deviation of the local neighborhood. This enforces local competition between adjacent features (adjacent in space as well as kernel dimension) in a feature map.

Krizhevsky et. al. [3] implement a similar module termed "local response normalization" with the following form:

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max 0, \frac{i-n}{2}}^{\min(N-1, \frac{i+n}{2})} (a^j_{x,y})^2 \right)^{\beta} \tag{1}$$

where $a^i_{x,y}$ is the output of applying kernel i at spatial location (x, y) then applying a ReLU non-linearity, and $b^i_{x,y}$ is the response-normalized output. The sum runs across "adjacent" kernel maps at the same spatial location, normalizing each value by a sum of adjacent (in kernel space) feature values. This similarly creates competition for large values in neuron outputs using different kernels.

There are a few differences between these normalizations and the windowed activations we implement in this work. While both of these modules consider adjacent kernel mappings at the same spatial location in their normalization, our activations solely modulate the signal according to the spatial neighborhood. Additionally, these works apply the local normalization after a rectifying non-linearity, whereas many of our functions reverse this order. Finally, these two modules mimic inhibitory behavior in neurons, inducing competition between neighboring activations. In our work we also attempt to model excitatory behavior of neurons, with some of our activations amplifying signal in proportion to local activity rather than inhibiting it.

2.2. Nonlinear Convolution

Our work can also be seen as an attempt to increase the expressiveness of the standard CNN layer by increasing the receptive field and gradient flow of the activation function through window-based operations. Some previous work aims to increase the expressiveness of the standard CNN layer by defining alternative, non-linear convolutions.

Lin et. al. [4] replace the standard linear convolution with an approximation of a general nonlinear function by appending a two-layer multi-layer perceptron (MLP) to each convolutional layer. Similarly, Zoumpourlis et. al. [10] approximate a nonlinear function using polynomial expansion. They implement a quadratic convolution using two sets of kernel weights: the first-order kernel contains the coefficients of the filter's linear part, while the second-order kernel contains the coefficients of quadratic interactions between two input elements. Finally, Kim et. al. [2] propose a non-linear convolutional filter as the negative square of subtraction between the image patch and kernel weights. They show that their filter results in a faster reduction in test error over the standard linear convolution for image classification on black-and-white and gray-scale images.

3. Technical Approach

Similar to how convolution is applied to raw pixels or feature maps, our window-based activations use PyTorch's native `unfold` and `fold` functions to retrieve patches of an activation map and stitch it back together. Although our proposed activations utilize patches or windows of feature maps the comparison to convolution ends there. Instead of performing aggregations per element, our activations utilize various window statistics to influence the final output. This allows the activation to mimic some of the dynamics seen in neuroscience and biology where "neurons" with low activations could impact high activations and vice versa.

We begin with an activation map from a convolutional layer, extract its patches so that each patch is non-overlapping and is only of size H x W (each channel will be separated into its own set of patches). Once the patches are extracted, an activation is ran on each patch individually. Once the activation functions have been applied, the filtered patches are then stitched back to the original dimensions and passed to the next layer of the network. We have found empirically that the window based activation functions are slower than element-wise activations. Window based activations average around 2-3 times slower than ReLU variants.

This implementation of window based activations differs from previous work in a few ways. First, we use specific window statistics (weighted sum of negative values, etc.) to influence the propagation of the feature patch values. Second, the window activation function applied to each patch can inspire competition or collaboration between values (excitatory vs inhibitory) depending on the window statistics used. Lastly, differential and non-differential functions can be used so long as the patch function applies a selective index to the original values (similar to ReLU) which is done for a few of our activation functions. These differences allow for a wider search of window based functions and open up new possibilities for biologically inspired networks. In our experiments we tested seven different activation functions described below.

NeLU (see Figure 2), for example, is meant to loosely resemble inhibitory neurons by draining portions of positive activations, potentially preventing them from firing. To accomplish this, NeLU takes a weighted sum of all the negative activations and adds that to each of the positive elements in the window. After the negative values have been added to the positive activations, only the activations that remain positive are allowed to pass through to the next layer. An interesting side-effect of NeLU when compared to ReLU is that although negative activations do not propagate forward (like ReLU), those negative activations will have a gradient so long as one activation in their window is positive. Although this is an area we are still working on, we believe this helps facilitate training of "dead neurons", which can occur frequently using ReLU. In NeLU, "dead neurons" would only occur with "dead patches" which seems far less likely.

Excitator (see Figure 2) is a similar activation to NeLU, though it takes a weighted sum of the positive values and adds it to the negatives. The positive values are then adjusted for the shift in activation proportional to the strength of their original signal. After the positive influence has been distributed, anything now positive is passed through. Although this doesn't allow negative neurons to have a gradient to train on, it reduces sparsity within the patch by weakening the negative signal of nearby activations (potentially flipping them to positive and allowing them to propagate).

Passive NeLU is the third proposed activation, and it behaves similar to NeLU again, with one exception, it allows

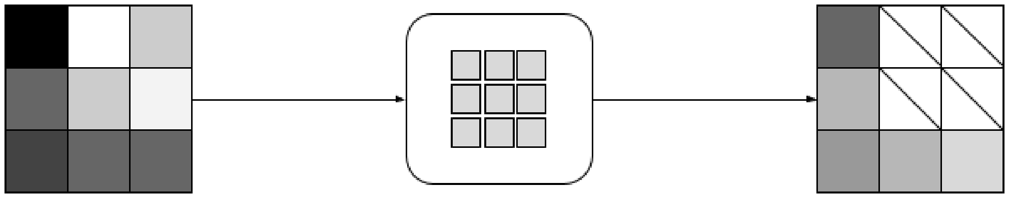

Figure 1. **Window-Based Activation Function** A generic model for the proposed set of window-based activation functions, where neighboring activations have different effects on each other.

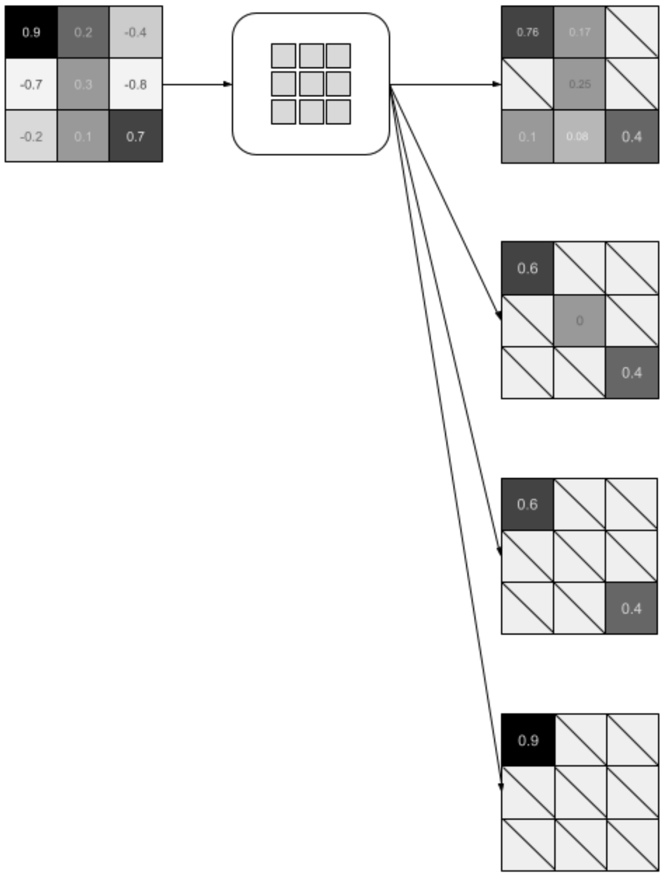

Figure 2. **Example Window-Based Activation Functions** An example input patch, and the corresponding outputs for the subset of activation functions: (1) excitator, (2) inhibitor, (3) nelu, and (4) max.

anything that was originally positive to propagate (even if the activation was turned negative after the redistribution of signals). Because of this interaction, Passive NeLU allows certain negative values to propagate forward and allows for gradients of non-propagated signals.

Inhibitor (see Figure 2) is our fourth activation and is similar to Excitator except it performs the equal redistribution of negative weight on the positive signals. This allows

Activation	Tiny IN	Caltech-101	CIFAR-100
ReLU	30.74	44.32	40.3
LeakyReLU	31.11	46.16	39.91
GELU	30.04	41.17	41.41
Max	23.45	37.79	43.33
MaxReLU	23.56	41.17	42.08
Softmax Threshold	29.35	46.62	39.55
NeLU	27.33	38.33	40.03
PNeLU	29.62	47.77	36.71
Inhibitor	27.28	42.70	37.58
Excitator	**34.27**	**47.47**	**47.37**

Table 1. Table showing the test accuracies of the baseline activations after 50 epochs of training vs the test accuracies of our activations. All results are trained on a custom ResNet-38 model with no pre-training. Excitator outperforms all baselines, in some cases dramatically as seen in CIFAR-100 where it achieves nearly 6 points higher than the leading baseline.

for negative signals to have gradients and creates a very sparse activation layer.

SoftmaxReLU applies a softmax to every value in the feature patch, then performs a ReLU like thresholding on the values. The threshold that allows values to propagate is a hyperparameter and we set ours to .25 for our experiments.

MaxReLU finds the maximum value within the patch and only allows it to propagate if it is above 0.

Max (see Figure 2) finds the maximum value within the patch and allows it to propagate regardless of its value.

4. Results

See Table 1 and Table 2

We tested all 7 activation functions on a custom implementation of ResNet-38 and a pre-trained EfficientNet-b0. We compared each of our activations with 3 baseline activations that are similar to ours: ReLU, LeakyReLU, and GeLU. Furthermore, we tested each of our activation functions on various different hyper-parameter settings including differing window sizes and influence weight. Each test was run on three different datasets: Tiny ImageNet, Caltech-101 and CIFAR-100. Accuracy is measured on the validation set and the task to solve for is image classification.

The results show that Excitator is a fairly strong activation function, with Excitator outperforming all baselines on the ResNet model. Although other activations did manage to outperform a number of the baseline activations – Excitator (a signal amplifier window activation function) seems to perform consistently better than all others. Krizhevsky et. al. [3] also experienced a large jump in classification scores during their experiments with competitive kernels.

Activation	Tiny IN	Caltech-101	CIFAR-100
ReLU	62.89	81.87	57.86
LeakyReLU	62.85	**82.95**	**58.76**
GELU	**63.39**	82.33	58.51
MaxReLU	57.03	72.89	50.12
SoftmaxReLU	59.02	79.26	53.98
Inhibitor v2	62.31	82.33	57.55
Excitator v2	61.97	82.18	56.84

Table 2. Results using a pre-trained EfficientNet-b0 (5.3M params) and baseline vs. our activations on TinyImageNet, Caltech-101, and CIFAR-100. Results shown are the test set accuracies using the best hyperparameters after 15 epochs of training.

An exciting area of future work would be to re-implement Krizhevsky et. al. and compare them with Excitator and other window based activations.

4.1. Experiments on EfficientNet

We also tested our activations using EfficientNet [9], a state-of-the-art CNN used for image recognition. EfficientNet uses Swish [6] as an activation function between most layers in the model. To test our activations, we substituted our activation function after the first convolutional layer in the model, with subsequent layers unchanged. We initialized the model using pretrained weights[1], swapped in our activation function (or a baseline activation function), then finetuned using the training set of each dataset.

Table 2 shows the results of our modified EfficientNet on three different datasets. GELU performs the best on Tiny ImageNet, while LeakyReLU performs the best on CIFAR-100 and Caltech-101. On Caltech-101, Inhibitor v2 outperforms ReLU and matches the performance of GELU. However, overall we do not see the same performance gains we did on our custom ResNet model using windowed activations. Further experiments might explore adding our activations in more and different places throughout the EfficientNet architecture, training for longer, and conducting a wider hyperparameter search. It's also plausible that training from a random initialization with our activations may work better than finetuning a pretrained model.

5. Conclusion

We have introduced a new class of activation functions, window-based activations, and shown that they can outperform state-of-the-art element-wise activation functions with proper training time, hyperparameter settings, and initializations. However, there is still an extremely large amount of work that can be done. Exploring trade-offs between

[1]https://github.com/lukemelas/EfficientNet-PyTorch

pretrained models and random initialization, more windo–based activations, and different types of window-based non-linearities are all possible and should be explored in future work.

References

[1] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009. 1

[2] Hyoseob Kim, Hojun Yoo, Jung Lyul Lee, and Seoungho Lee. Convolution layer with nonlinear kernel of square of subtraction for dark-direction-free recognition of images. *Mathematical Models in Engineering*, 6(3):147–159, 2020. 2

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 1, 4

[4] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 2

[5] Pitts W. McCulloch, W.S. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. 1

[6] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 4

[7] Terrence J. Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117(48):30033–30038, 2020. 1

[8] Dee U. Silverthorn. *Human Physiology: An Integrated Approach.* Pearson/Benjamin Cummings, San Francisco, 2007. 1

[9] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 4

[10] Georgios Zoumpourlis, Alexandros Doumanoglou, Nicholas Vretos, and Petros Daras. Non-linear convolution filters for cnn-based learning. In *Proceedings of the IEEE international conference on computer vision*, pages 4761–4769, 2017. 2