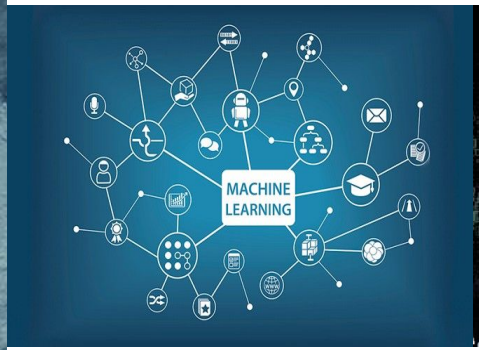
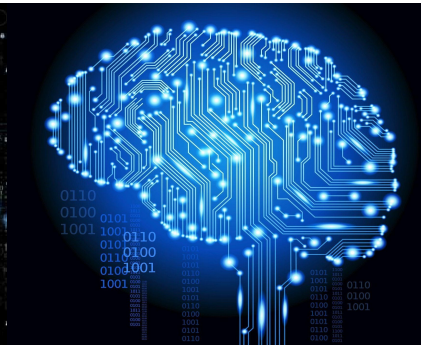
A Deeper Understanding of Activation Functions

ReLU vs. Swish



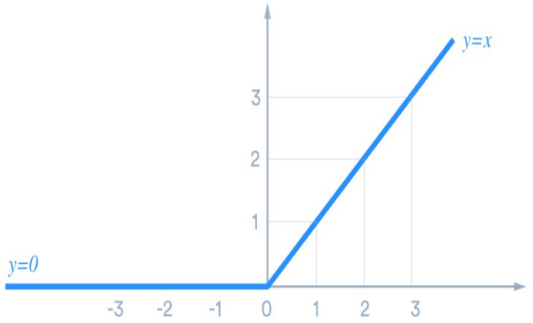

Activation Functions

- A function that takes in the weighted sum of all of the inputs from the previous layer and then generates and passes an output value (typically nonlinear) to the next layer
- By adding an activation function, adding layers has more impact
  - Stacking nonlinearities on nonlinearities lets us model very complicated relationships between the inputs and the predicted outputs.
- In brief, each layer is effectively learning a more complex, higher-level function over the raw inputs

- Swish Activation
  - new self-gated activation function discovered by researchers at Google
  - proposed to replace ReLU
- ReLU Activation
  - the most successful and widely-used activation function

ReLU Activation Function

- Rectified Linear Unit
- f(x) = max(0,x)
- default activation function: easier to train and achieves better performance
- piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero
- unboundedness; desirable because it avoids saturation, where training is slow due to near-zero gradients; overcomes vanishing gradient problem
    - models to learn faster and perform better
- units are nearly linear → preserve many of the properties that make linear models easy to optimize with gradient-based methods & generalizes well
- All negative values become 0 immediately, which decreases model's ability to fit or train from data properly
    - Affects resulting graph by not plotting negative values appropriately
- Model Sparsity
    - concise models that often have better predictive power and less overfitting/noise
    - neurons are actually processing meaningful aspects of the problem



Swish Activation Function

- $f(x) = x \cdot \text{sigmoid}(x)$
- Swish is *smooth* (it does not have sudden changes of motion or a vertex)
    - smoothness helps optimize and generalize the neural network
- *non-monotonic*, meaning that there is not always a singularly and continually positive (or negative) derivative throughout the entire function
    - the non-monotonicity property of Swish distinguishes itself from most common activation functions
- Swish is unbounded above and bounded below
    - unboundedness; desirable because it avoids saturation, where training is slow due to near-zero gradients

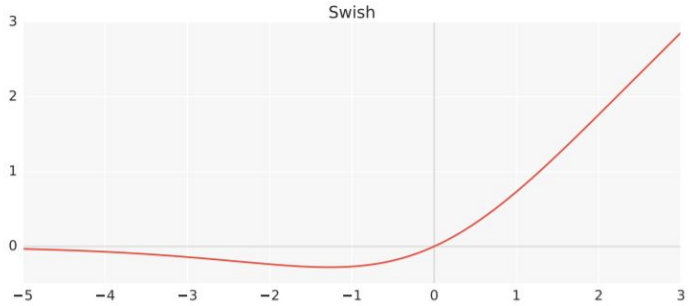
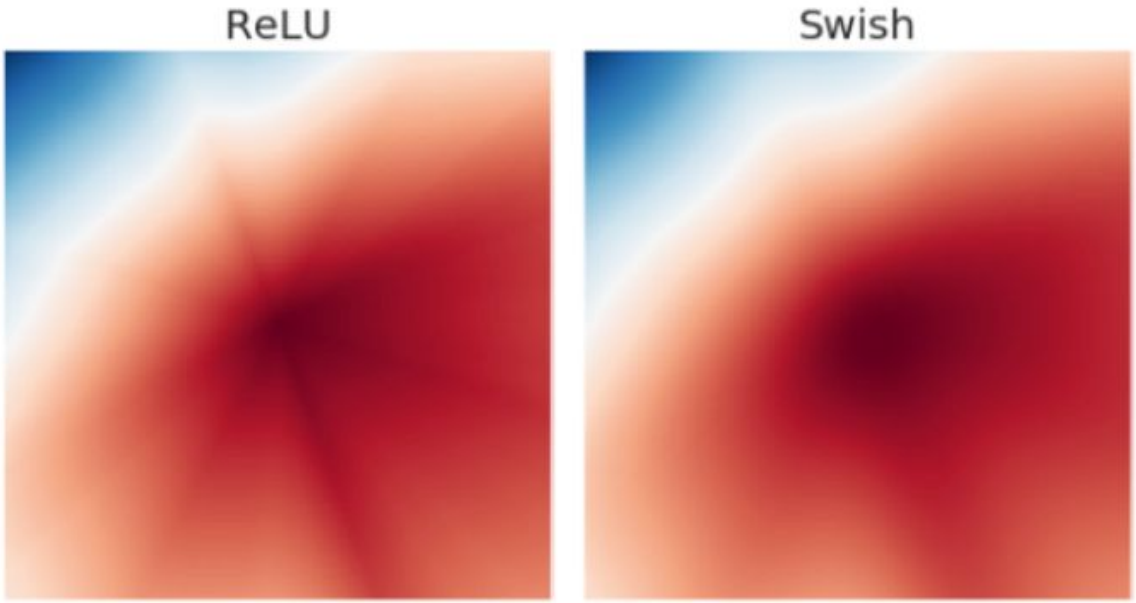


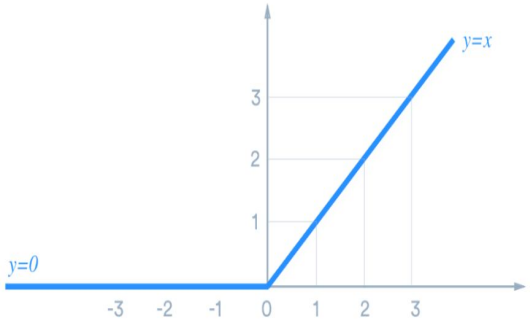
Figure 1: The Swish activation function.



In the output landscapes above, it is obvious that ReLU's network output landscape has distinctive sharp and jarring regions of similar magnitudes because of its non-smooth nature, whereas the Swish network output landscape is much more smoother.

Swish vs. ReLU Activation Functions

- implication that the gradient preserving property of ReLU may no longer be a distinct advantage in modern architectures
- properties of Swish being unbounded above, bounded below, non-monotonic, and smooth are all advantageous.
- Swish differs from ReLU because it produces negative outputs for small negative inputs due to its non-monotonicity
    - increases expressivity and improves gradient flow, important considering that many preactivations fall into this range
- ReLU empirically underperforms Swish, but ReLU benefits from a comparatively simpler theoretical analysis due to its piecewise linear form



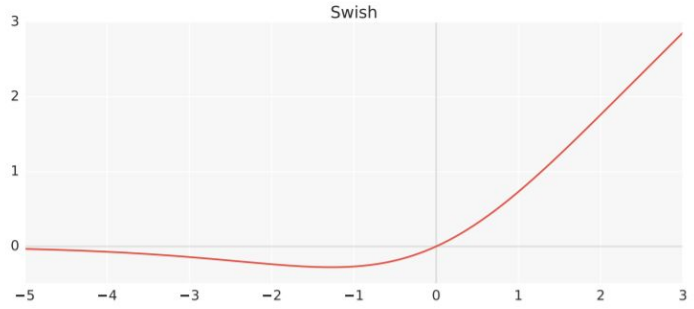
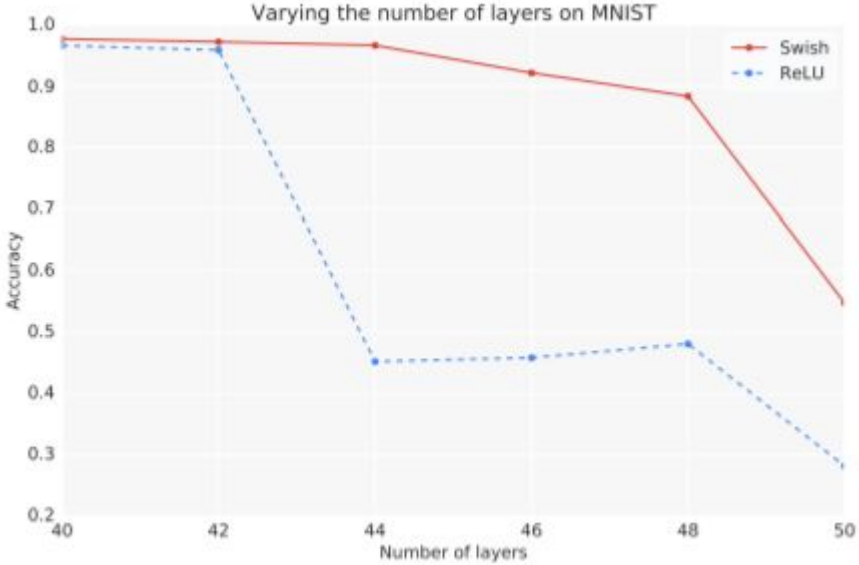


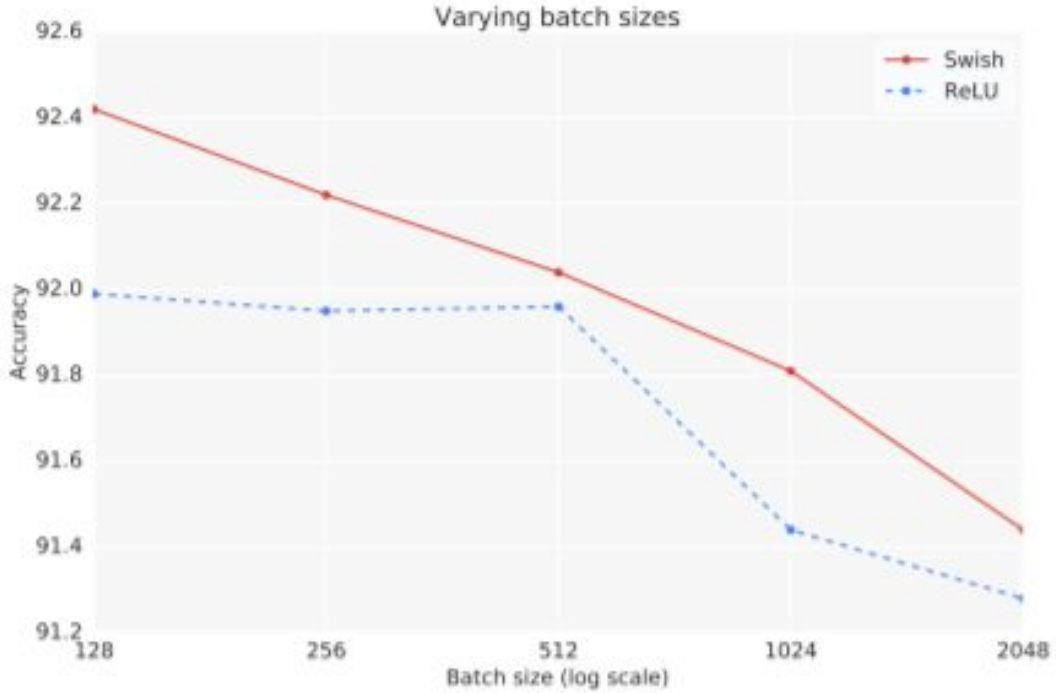
Figure 1: The Swish activation function.



The test accuracy when changing the number of layers in fully connected networks with different activation functions on MNIST.

- results plotted are the medians of 3 runs
- in experiment, both activation functions achieve similar performances up to 40 layers
- Swish outperforms ReLU by a large margin in the range between 40 and 50 layers when optimization becomes difficult.
- In very deep networks, Swish achieves higher test accuracies than ReLU.

This result challenges the conventional wisdom that ReLU performs well because it does not squish gradients; Swish is able to train deeper networks despite having gradient squishing properties.



Swish outperforms ReLU on every batch size, suggesting that the performance difference between the two activation functions remains even when varying the batch size.

Baselines	ReLU	LReLU	PReLU	Softplus	ELU	SELU
Swish > Baseline	9	8	6	7	8	8
Swish = Baseline	0	1	3	1	0	1
Swish < Baseline	0	0	0	1	1	0

The above table demonstrates how many times Swish outperforms, is equivalent to, or underperforms each outlined baseline activation functions at 9 experiments.

Average Validation Score of ReLU vs. Swish (N)

- Took Neha's top performing model (w/ Swish) & ran 5 training sessions & took average
- Swish Avg: .96198
- ReLU Avg: .9568

Swish Values: 9616, .9630, .9606, .9623, .9624
ReLU Values: 9575, .9586, .9567, .9580, .9532

```
learning_rate = 0.003
epochs = 50
batch_size = 4000
validation_split = 0.2
```

```
# Define the first hidden layer.
model.add(tf.keras.layers.Dense(units=32, activation='swish'))
```

```
# Define the first hidden layer.
model.add(tf.keras.layers.Dense(units=32, activation='relu'))
```

Average Validation Score of ReLU vs. Swish (J)

- Took Josh's top performing model (w/ ReLU) & ran 5 training sessions & took average
- Swish Avg: .97456
- ReLU Avg: .9715

Swish Values: .9718, .9730, .9738, .9789, .9753
ReLU Values: .9711, .9727, .9733, .9706, .9698

```
#first layer
  model.add(tf.keras.layers.Dense(units= 65, activation= 'swish'))

#second layer
  model.add(tf.keras.layers.Dense(units= 35, activation = 'swish' ))

#third layer
  model.add(tf.keras.layers.Dense(units= 27 , activation = 'swish'))
```

```
#first layer
  model.add(tf.keras.layers.Dense(units= 65, activation= 'relu'))

#second layer
  model.add(tf.keras.layers.Dense(units= 35, activation = 'relu' ))

#third layer
  model.add(tf.keras.layers.Dense(units= 27 , activation = 'relu'))
```

```
learning_rate = 0.005
epochs = 50
batch_size = 4000
validation_split = 0.25
```

Findings

- Swish consistently matches or outperforms ReLU on deep networks applied to a variety of challenging domains such as image classification and machine translation
- improves the training of deep networks & robustly outperforms ReLU with different batch sizes
- has the properties of one-sided boundedness at zero, smoothness, and non-monotonicity, which may play a role in the observed efficacy of Swish
- Experiments used models and hyperparameters that were designed for ReLU and just replaced the ReLU activation function with Swish; even this simple, suboptimal procedure resulted in Swish consistently outperforming ReLU and other activation functions.
- expect additional gains to be made when these models and hyperparameters are specifically designed with Swish in mind
- The simplicity of Swish and its similarity to ReLU means that replacing ReLUs in any network is just a simple one line code change.