Activation Functions in Text & Image Classification

2024-12-15

Contents

Abstract

The project aims to examine the performance of three activation functions ReLU, Leaky ReLU and Swish on text processing and image processing tasks by using the Gated Recurrent Unit (GRU) for text and Convolution Neural Network (CNN) for images. The analysis of our project focuses on the Fetch 20 News dataset and for text classification and CIFAR 10 dataset for image classification. Experiments are developed with different configuration to assess the performance of all three activation functions, which have been inspired by Ramchandran (Ramachandran et al., 2017) work, to see whether the Swish function consistently outperforms the traditional ReLU and its hand designed activation functions like Leaky ReLU function on different architectures in terms of accuracy and loss and if it Swish function is a better alternative to use in deep learning architectures going forward. As per our results Swish outperformed ReLU and Leaky ReLU on most of configurations but is not as consistent as the Author in his paper claims it to be.

Introduction

Goals

We have developed two models, one for classification of text and the second for classification of images. The models will run on each of the three activation functions to assess and evaluate the performance. Finally, we will be testing the claims of the Author, whether the Swish outperforms the ReLU and its relatively new variant Leaky ReLU.

Text Classification

The goal here is to classify as accurately as possible the text from “fetch 20 news group” dataset into one of twenty predefined categories in the dataset by slightly making the model more complex in each of configuration, which have been mentioned in the dataset description. We are comparing the performance of activation of three activation function (ReLU, Leaky ReLU and Swish) in a Gated Recurrent Unit (GRU), which is a type of recurrent neural network to predict whether the model is correctly able to classify the text into right categories, making it an excellent source for analyzing the different activation functions and configurations.

Image Classification

The primary objective of this was to train convolutional neural networks (CNNs) on the CIFAR-10 dataset to evaluate the performance of three activation functions: ReLU, Leaky ReLU, and Swish across varying levels of model complexity. The experiment setup was carefully designed to enable fair comparisons, ensure efficient training, and provide insights into the behavior of activation functions under different circumstances.

Activation Functions and Neural Network Architectures

Activation Functions

- **ReLU (Rectified Linear Unit):** Defined as $f(x)=\max(0,x)$, ReLU introduces sparsity and accelerates convergence due to its simplicity. It served as the baseline activation.
- **Leaky ReLU:** A variant of ReLU, defined as $f(x)=x$ if $x > 0$, and $f(x)=\alpha x$ for $x \leq 0$ ($\alpha=0.01$). It mitigates the "dying ReLU" problem by allowing gradients to flow for negative values.

- **Swish:** Defined as $f(x)=x\cdot\text{sigmoid}(x)$, Swish introduces non-monotonicity, which improves gradient flow, making it particularly effective for deeper architectures.

Neural Network Architectures

A Gated Recurrent Unit (GRU) is a special type of Recurrent Neural Network which is basically used in processing of sequential data, making it a suitable choice for text processing tasks. It also addresses the problems faced by the traditional RNN networks, for example the problem of vanishing gradient. To regulate the flow of information, this network architecture brings into play a gating mechanism. This gating mechanism makes it very effective for the tasks such as sequential dependencies and natural language processing (NLP). The two gates GRU uses are the update gate and reset gate. The update gate decides regarding the past information that has to be carried forward, trading off between remembrance of the important and forgetting the irrelevant data. While the reset gate, works by deciding how much previous state should be ignored while computing the current state. Both of these gates together work to capture the relevant long- and short-term dependencies in the data. This balance between the efficiency and performance makes the GRU relatively powerful for handling of data.

A Convolutional Neural Network (CNN) is a specialized deep learning architecture designed to process grid-like data such as images. It uses convolutional layers with small filters to automatically extract spatial features, such as edges, textures, and shapes by sliding across the input. Activation functions like ReLU, Leaky ReLU, or Swish introduce non-linearity which enables the network to learn complex patterns. Pooling layers (such as: MaxPooling) downsample feature maps to reduce computational complexity and control overfitting. Additional components like Batch Normalization stabilize training, while Dropout prevents overfitting by randomly deactivating neurons. The output from convolutional layers is often passed through Global Average Pooling (GAP) or flattened before feeding into fully connected or dense layers. A final softmax layer assigns class probabilities for the classification tasks. CNNs' ability to hierarchically learn features from raw data makes them highly effective for image classification, object detection, and other computer vision tasks.

Background (Literature Review)

ReLU and its variants have been considered an integral part of modern deep learning architecture and have influenced the performance of neural networks related to image and text processing. This review describes how ReLU and its modified versions have enhanced the efficiency in computation, representation learning, and generalization in both domains. The introduction of the ReLU is a huge advancement in training neural networks toward computer vision tasks. Et al.'s (2015) paper describes the advantages of Parametric ReLU (PReLU) over classic ReLU, particularly in the context of their image classification work using the ImageNet dataset. It provides adaptive fitting to improve data fitting by about 1.2 percentage points, resulting in a top-5 error rate decrease from 13.34% to 12.75%. This improvement is mainly attributed to the capability of PReLU to avoid the "dying ReLU" problem so that active gradients can pass through the deeper layers of convolutional neural networks (CNNs). These advantages are particularly availed in very deep networks that possess 30 or more layers, and standard ReLU usually fails to sustain gradient flow through them.

Further improvements, such as ELU, have started outperforming ReLU on both convergence speed and accuracy on image recognition tasks. For example, Clevert et al. (2016) showed that the single-layer ELU network had achieved the state-of-the-art performance on datasets such as ImageNet with better training efficiency and final accuracy compared to its ReLU counterpart. Furthermore, dynamic and piecewise-linear activation functions have enlisted as some of the good tools for performance improvement in image processing. To illustrate, Rane et al. (2024) termed their work Adaptive Activation (AdAct) because it employs piecewise-linear components that

outperformed the static ReLU activations. This innovation enabled CNNs to perform refined feature extraction, more attuned to demands like object recognition and segmentation.

While ReLU has revolutionized image processing, it has not proved very effective when applied to text processing. The reason is, unbounded nature of ReLU takes benefits for visual tasks, whereas in text models, where sparsity and smooth gradient flow are important, it creates instability. Therefore, ReLU variations have been created and adapted specifically for sequential and textual data. Banerjee et al. (2020) presented multi-phase ReLU functions. The original ReLU has been modified to allow it to offer multi-output provisions. This variant has, however, shown its potency especially in RNN models, as well as in applications like text classification and sentiment analysis. It added flexibility that made improvements in convergence for models and overall accuracy over RNNs built using traditional ReLU. Piecewise-linear units have also formed part of the different architectures applied in text processing to capture better the complex semantics in textual data. They have proven excellent in making it possible to approximate non-linear mappings with a very high degree of accuracy, and they are hence indispensable for applications such as language modeling and machine translation, where the standard ReLU often proves inadequate in modeling such sophisticated patterns.

The leaky ReLU, or LReLU, turns out to be a significant improvement in deep learning because it solves several drawbacks that are linked to ReLU. The most common drawback of the ReLU is referred to as 'dying ReLU' where it fails to propagate the back gradient for negative values thereby leading to the gaining of uphealthy neurons during the training process Robust Convolutional Neural Networks with leaky ReLU convolutional layers and a small amount of negative bias could avoid this problem as it enables the efficient and effective backpropagation of the neural networks even for big data irrespective of the noise. The current review is devoted to the relations between LReLU and newer LReLU-memory models, connections, applications, and future developments. For the given data complexity, dynamic adaption of the activation slope is done by introducing a learnable parameter ,α , hence the Leaky ReLU so termed as LeLeLU. LeLeLU outperformed ReLU by 2.53% improvement in classification accuracy for images and tabulated data of a number of fields. Absolute Leaky ReLU (ALReLU): This variant uses absolute values for negative gradients in order to alleviate vanishing gradient problems. ALReLU has reported remarkable performance improvements in medical imaging and text classification tasks.

In the cases of generative neural networks, adversarial network teaching models, LReLU has contributed to greater stability and higher quality of samples used in training. The use of sentence embedding available in LReLU has resulted in easy training, improved high dimensional space handling in particular cases of GAN architectures. Such studies found that LReLU has consistently been rated higher than ReLU, especially when LReLU is used in speech or other noisy datasets. Because it has a less steep negative slope, LReLU provides better convergence and representation learning since it is capable of allowing gradient flow even through negative inputs. In addition, LeLeLU and AlReLU variants are even more flexible and powerful as they provide closer fits of activation functions to data. However, even with those benefits, LReLU and its relatives still have to deal with several challenges: The fixed alpha - the actual value of Leaky ReLU is a well-differentiated scalar alpha –also may not need making it an optimization limitation on generalization, while mapping more advanced LReLUs on neural networks models imply a greater computation burden due to additional learning parameters. Replacement or expansion of LReLU efficiency to more advanced models: Transformers, deep GANs will take some time. Creating special types of activation functions is still an open problem, such as multi-modal data fusion or functions that work only for areas of neuroscience Adaptive insertion of the activation methods might be an interesting topic for future research. It may be done with meta-learning or, for example, directional evolutionary strategies to modify activation parameters according to the task

Meanwhile, Swish function has shown a superior performance in image processing problems. Ramchandran, introduces the Swish function that is optimized via reinforcement learning show the

advantages over RELU and Leaky RELU in different deep learning architectures (Ramachandran et al., 2017). Different datasets like the Image Net showcased a consistent improvement in the accuracy metrics in the different model like Inception ResNet and Mobile NasNet A where the swish activation functions performed better by 0.9%.

Further insight in to the Swish's non monotonic bum allows better flow of gradient for the negative inputs, improving the convergence and learning rates. For instance Chieng, in his research paper floats the flatten T swish (FTS), one of the swish variant's, which addresses the ineffectiveness of RELU to make utilization of the negative inputs. The FTS swish function showed a quicker convergence and higher accuracy compared to RELU in the image classification using the MNIST datasets (Chieng et al., 2018). In the domains of semantic segmentation and object recognition Mercioni, introduced another variant of Swish Activation function known as P-Swish, a parametric of Swish variant. This variant shows a significant improved metrics like intersection over union and validation accuracy on datasets that include 2018 Science Bowl and CIFAR 10. This highlights Swish's adaptability in computer vision applications (Mercioni & Holban, 2020).

The capabilities of Swish activation function extend further to natural language processing (NLP) as well as Eger, in his research conducted a detailed comparative analysis of swish and 8 other activation functions including RELU and Sigmoid, his results show that the Swish outperforms other in a consistent manner in text classification, sentiment analysis and machine translation (Eger et al., 2019). The Swish's success in his paper is attributed to the non monotonic and smooth characteristics which facilitate the learning of complex patters by enhancing the gradient flow in the text data. Furthermore, Swish has also been integrated into attention mechanisms of transformer model including but not limited to BERT and GPR. The results of experiment conducted showcase the improved BLEU scores in machine translation problems and generalization in summarizing the text, validating reliability of Swish for NLP models (Ramachandran et al., 2017).

Swish's performance and adaptive metrics are further improved by different variant for example in case of Flatten T swish, t integrates a learnable threshold (T), allowing the network in propagating the negative inputs in an effective manner . This further enhances the predictive behavior and reduces the complexity for computation during the training process (Chieng et al., 2018). While the P Swish introduces different learnable parameters refining the dynamics of the function even more. This particular variant showed the maximum accuracy in the problems that required feature extraction, proving its suitability for both vision and language tasks (Mercioni & Holban, 2020). Swish and its variants provide quite a robust option to choose compared to the traditional activation functions. Future research into this function should be focused in optimizing implementations to diminish the computational costs and broaden the applications for the use of Swish activation function.

Description of Dataset

Text classification (Fetch 20 Newsgroup)

Fetch 20 Newsgroup dataset is quite popular dataset that is used as a benchmark in particularly text classification, natural language processing and machine learning. The dataset contains 20000 newsgroup documents obtained from twenty different categories which are distributed evenly. Every document belongs to one of the predefined categories, making this dataset a multiclass classification task. The 20 categories are organized under six broader themes, which are as follows (all 20 are mentioned in the appendix):

1- Computers (5 Categories)
2- Recreation (4 Categories)

3- Science (4 Categories)
4- Miscellaneous (1 Category)
5- Politics (3 Categories)
6- Religion (3 Categories)

This dataset brings a challenge in classification problems because of overlapping of vocabularies among the categories. For example, "*comp.sys.ibm.pc.hardware*" and "*comp.sys.mac.hardware*" are two categories which share technical terms, making distinguishing between them difficult for the model. The overlapping attribute of this dataset make it a great testing problem for assessing the ability of different machine learning models, to get an insight into the sequential relationship and context of the different categories.

Every document is in the dataset in the form raw text, which contains headers, footers, and quotes. These elements bring in noise into this dataset which need to be removed via the preprocessing to make data clean for the model. The documents also are of differing lengths and contain diverse linguistic English language data with abbreviations, specific jargons for a domain, informal language making it difficult to generalize in an effective manner. This defined structure and variety in the data makes it a comprehensive for use in exploring the performance on text classification problems.

Image Classification (CIFAR-10)

The CIFAR-10 dataset is a widely used dataset in computer vision, specifically for image classification tasks. It consists of 60,000 color images, each measuring 32x32 pixels, with 50,000 images designated for training and 10,000 for testing. Each image contains three color channels (RGB) and belongs to one of 10 distinct classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, or truck. The dataset is well-balanced with an equal number of images per class.

Although the images are small in size, CIFAR-10 presents a meaningful challenge due to variations in object orientation, lighting, and positioning. Its balanced class distribution and compact nature make it computationally efficient, that allow researchers and practitioners to quickly train and test models. This makes the dataset especially valuable for evaluating deep learning techniques such as Convolutional Neural Networks (CNNs), testing different activation functions and fine-tuning model hyper parameters.

CIFAR-10's simplicity and diversity make it perfect for experimentation, allowing researchers to explore model performance, test regularization strategies, and optimize neural network architectures. Despite its straightforward design, it continues to be an essential dataset for making advancements image classification algorithms in machine learning research.

Experiment Design

Text Classification

- The "Fetch 20 Newsgroup Dataset" is directly loaded from the sk.learn datasetsts library of python. With text the data also contains a lot of headers, footers and quotes, which are removed to remove the noise from the dataset and make it compatible for the GRU model.
- The dataset is quite large, with respect to computational resources and time, we have limited our sample size to 5000. The data is divided into feature x, which has labels and y which has all the relevant 20 categories.
- The data that is obtained is in the form of text, and before proceeding with the development of model it has been preprocessed, making it suitable for the model.

- For this dataset we opted for SpaC's NLP pipleline, which tokenizes the text and removes the unnecessary words, punctuation and non alphabet characters. This basically ensure removal of any irrelevant data that might influence the model later on.
- The tokenized words form the Keras library is used to vectorize them into numerical form. This step in the process is done to convert the words into integers, based on the frequency.
- To make sure that the input dimensions are uniform, the sequences have been padded to a fixed length of 100 tokens. Encoding of text labels into numerics is done via the Label Encoder, as neural network models work on numerical data for processing. The dataset is divided into training and testing splits of 80% to 20%.
- The architecture for the model is designed with the Gated Recurrent Unit (GRU) layers, it is quite suitable to handle sequential data like text.
- The three configurations for text processing (shallow, Medium, Deep) have been defined with different parameters to run and compare the performance of different activation functions.
- The parameters include GRU layers, Hidden Units, Learning rate, Epochs and the dropout rate. The final layer, which is a dense layer contains 20 neurons of SoftMax function that basically output the probabilities of all twenty categories.
- The model for three activation functions we are comparing, is trained with Adam's optimizer. Adam's optimizer basically continues to adapt the learning rate while training for convergence in an efficient manner.
- For multi classification problems categorical cross entropy function is incorporated into the model, each configuration is trained for a certain no of epochs with a fixed batch size of 128. This ensures that the computer resources available are utilized as efficiently as possible.
- All three activation functions are trained, after training on 80% of data the model is evaluated and assessed on the testing data. The accuracy and loss function visualization aid and help to explain the performance across the epochs. The best configuration (which is based on highest accuracy) is selected, followed by Actual predicted categories for the text data.
- The qualitative analysis give insight into the generalization performance of the models and handling of category specific data.
- Lastly, the results are analyzed to assess the claims of author, if swish outperforms other activation function, with increasing model complexity on text classification problems.

Image Classification

The models were implemented using TensorFlow/Keras, with modularity in mind. Each activation function was applied using a common model template, which allowed for the same architectural configurations to be tested across ReLU, Leaky ReLU, and Swish. The models consisted of convolutional layers for feature extraction, followed by Batch Normalization to stabilize training and MaxPooling for spatial downsampling. For deeper configurations, Global Average Pooling (GAP) replaced traditional flattening to reduce parameters and improve generalization, particularly as model complexity increased. Fully connected dense layers were added after GAP, providing the capacity for high-level feature learning.

Regularization techniques such as Dropout were consistently applied after convolutional layers, with dropout rates increasing as the models grew deeper (e.g., 20% in Shallow, 40-50% in Deep and Deeper configurations). This controlled overfitting, especially in Swish-based models where smooth gradients can sometimes increase the risk of overfitting. The Adam optimizer was used to train all models due to its adaptive learning rate capabilities, ensuring efficient convergence. The learning rate was carefully tuned,

set to 0.001 for the Shallow and Medium configurations, and lowered to 0.0005 for the Deep and Deeper models to stabilize training in highly complex networks.

The training process involved running each configuration for 30 epochs, with early stopping applied based on validation loss. Training and validation curves for accuracy and loss were plotted for all models to observe learning dynamics, convergence behavior, and any overfitting trends. Swish, for example, exhibited smoother validation loss curves, whereas ReLU and Leaky ReLU often showed fluctuations, particularly in deeper networks.

A key element of the experiment was the use of sample predictions to qualitatively assess model performance. For each activation function, predictions on a sample of test images were visualized alongside their true labels, providing insight into how well the models recognized different classes. This visual assessment highlighted Swish's superior ability to generalize, as it produced fewer misclassifications compared to ReLU and Leaky ReLU.

By implementing a consistent structure across all models in the code, the experiment ensured that the results were directly comparable. This design enabled a detailed comparison of ReLU, Leaky ReLU, and Swish activation functions, and focused on their ability to optimize convergence, stabilize gradients, and improve generalization across varying complexities in the network.

Configurations for Testing

Text Classification

Shallow Configurations:

- A single GRU layer with 32 hidden units
- An embedded layer for representing input (embedding dimension = 128)
- Dropout rate of 20%
- A dense layer with the softmax activation for classification of text into one of twenty categories
- Learning rate of 0.01
- Trained for 5 epochs

Medium Configuration:

- Two GRU layers with 64 hidden units in each of them
- An embedded layer for representing input (embedding dimension =129)
- The initial layer of GRU returns the input sequences into the second layer of GRU
- Dropout rate of 30% (moderate regularization)
- A dense layer with the softmax activation for classification of text into one of twenty categories
- Learning rate of 0.01
- Trained for 10 epochs

Deep Configuration

- Three GRU layers with 128 hidden units in each
- An embedded layer for representing input (embedding dimension =129)

- Each layer in the processes the sequential data, with the final layer outputting the non sequential data
- Dropout 40% (stronger regularization)
- A dense layer with the softmax activation for classification of text into one of twenty categories
- Learning rate 0.001 (more training time)
- Trained for 15 epochs

The temporal dependencies were captured in an effective manner by making the use of the sequential layers of the GRU model, which handle the text classification problem at hand, across all the configurations. The probabilities for all the twenty categories of the news data set were produced by the final layer that consisted of 20 neurons. The model was optimized using he Adam optimizer by tuning the learning rate (0.01 & 0.001). The model for all three activation functions was trained with the use of categorical cross entropy loss and batch size of 128 which was similar among all three configurations for text processing while the epochs and dropout percentage were increased to assess the incremental difference by tuning different parameters (goal of project). To further assess the impact of parameter tuning, accuracy and loss curves have been generated from the code for all three functions. Furthermore, a subset of test text samples are displayed with the true and predicted labels, giving a clear assessment of the models performance for each function.

The rationale behind opting for the above three GRU based configurations (shallow, medium and deep is based on the balancing of model's complexity, availability of computational resources, efficiency and its ability of learning of complex patterns in news dataset. Shallow configuration which ha single layer of GRU and least hidden units acts a baseline model to asses the performance of the activation functions with minimum model complexity to see how (subjective) good enough it is to learn the short term dependencies in the textual data. Followed by medium configuration, it has two GRU layers and double the hidden units as first layer, introduces a hierarchy in the structure which allows the model to learn features of intermediate level that the model with shallow configuration might not be able to learn. This configuration tries to balance the simplicity and generalization ability. Lastly, the deep configuration with three layer of GRU and four times hidden units compared to the shallow configuration is built to understand complex relations and the long term dependencies in the text data, making it ideal for understanding the complex problems with overlapping of vocabulary. The dropout regularization has been increase by 10% in every successive configuration to ensure the deep model does not overfit due to high capacity. Additionally, use of three different activation function add an element of non linear transformations across the three configuration which provide insight into the learning process and the performance of the models.

Image Classification

To evaluate these activation functions, the experiment utilized four model configurations of increasing complexity: shallow, medium, deep, and deeper architectures. Each configuration was designed to progressively increase the number of layers, filters, and regularization techniques.

In the **shallow configuration**, the model consisted of 2 convolutional layers with 32 and 64 filters. Batch Normalization was applied after each convolutional layer to stabilize learning and MaxPooling reduced spatial dimensions. A Dropout layer with a rate of 20% was added to prevent overfitting, followed by Global Average Pooling (GAP), which efficiently reduced parameters by replacing traditional flattening. The final dense layer had 128 neurons with ReLU activation, followed by the softmax output for classification. This configuration served as the baseline model for comparison. It allowed us to test how well activation functions performed on a simple, low-capacity network where overfitting is less likely, and training

dynamics are more stable. By limiting the number of layers and filters, this configuration ensured fast training and provided insights into the activation functions' ability to capture basic features.

The **medium configuration** increased the model complexity by adding a third convolutional layer with 128 filters. The dropout rate was increased to 30% and a dense layer with 256 neurons was added to better learn high-level features. The inclusion of an additional layer allowed for more complex feature extraction while maintaining regularization to prevent overfitting. The medium configuration was designed to explore whether increasing depth and filter size improves performance without introducing instability or overfitting. Adding an extra layer allowed the network to model intermediate-level features, such as edges, textures, and shapes which are critical for accurate classification. This configuration tested how activation functions handle moderately complex networks.

The **deep configuration** extended the architecture to 4 convolutional layers with filters of 32, 64, 128, and 256. This model introduced a dropout rate of 40% and a fully connected dense layer with 512 neurons. The deeper network allowed for more hierarchical feature learning, which tested the ability of the activation functions to handle increased depth and complexity. The deep configuration aimed to fully exploit the potential of the activation functions by increasing model depth and complexity. Deeper networks are capable of capturing more intricate, hierarchical features, such as object parts and combinations of patterns. This setup tested the ability of the activation functions to maintain smooth gradient flow, prevent vanishing gradients, and converge effectively.

The **deeper configuration** pushed the model to its limits, with 6 convolutional layers and filter sizes progressively increasing to 32, 64, 128, 256, 512, and 512. Regularization was heavily applied using Batch Normalization and dropout with a rate of 50% to prevent overfitting. A Dense layer with 1024 neurons was added before the softmax output layer. This configuration evaluated the activation functions in extremely deep networks, where maintaining gradient flow and stable convergence becomes challenging. The deeper configuration was designed to evaluate the activation functions under extreme depth, where training stability, gradient flow, and convergence often become major challenges. The rationale for including this model was to test whether activation functions like Swish, which excel at smoothing gradients, could outperform ReLU and Leaky ReLU in very deep networks. Additionally, this configuration highlights the importance of regularization and architectural choices in preventing overfitting and ensuring generalization.

All models were trained using the Adam optimizer to minimize the categorical cross-entropy loss. The learning rate was set to 0.001 for the shallow and medium configurations and 0.0005 for the deep and deeper models to ensure stability during training. A batch size of 128 was used across all configurations to balance computational efficiency and performance. Additionally, early stopping was implemented to monitor validation loss and terminate training when overfitting was detected.

The performance of each model was evaluated using test accuracy and test loss as the main metrics. To gain further insights, training and validation curves were plotted for accuracy and loss over epochs, allowing for an analysis of learning dynamics, convergence behaviour, and potential overfitting. A visual assessment was also performed by displaying a sample of test images with their true labels and predicted outputs, providing qualitative insights into the model's performance on different activation functions.

Results

Text Classification

Activation Function	Shallow	Medium	Deep
ReLU	31.2%	42.5%	26.5%
Leaky ReLU	30.4%	43.4%	32.6%
Swish	32.8 %	39.8%	34.2%

Table 1

ReLU

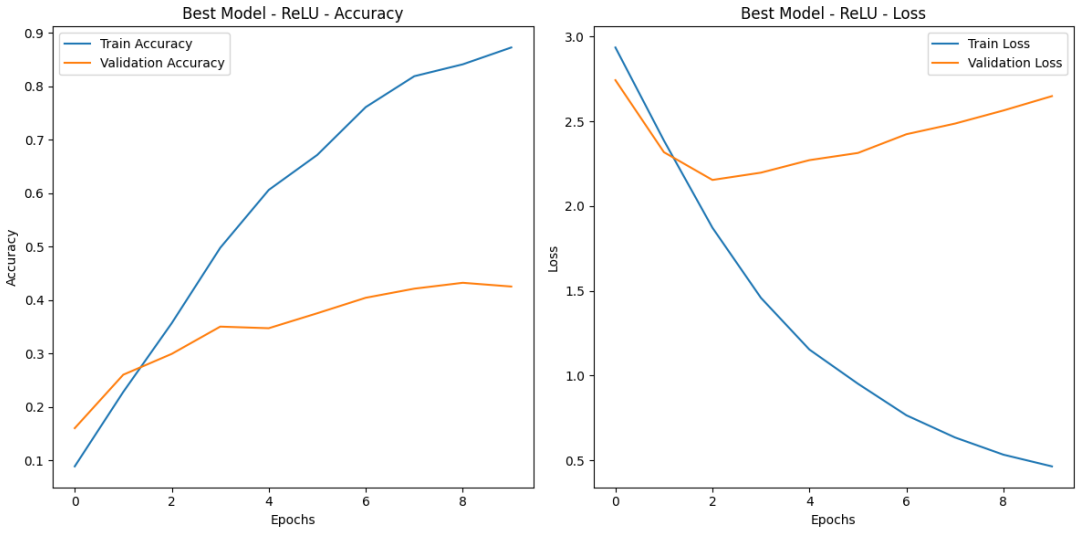

Figure 1: Accuracy and Loss Curves for ReLU

```
Sample Predictions:
Text:

I am sure some bashers of Pens fans are pretty confused about the lack
of any kind of posts about the recent Pens massacre of the Devils. Actually,
I am  bit puzzled too and a bit relieved. However,
Actual Category: 9
Predicted Category: 9
--------------------------------------------------
Text: My brother is in the market for a high-performance video card that supports
VESA local bus with 1-2MB RAM.  Does anyone have suggestions/ideas on:

  - Diamond Stealth Pro Local Bus

  - Orchid Farenh
Actual Category: 14
Predicted Category: 14
```

Figure 2a: Predicted vs Actual Categories (ReLU)

```
Text:



       Finally you said what you dream about. Mediterranean???? That was new....
       The area will be "greater" after some years, like your "holocaust" numbers.




              *****
       Is't July in USA now?????
Actual Category: 1
Predicted Category: 4
```

Figure 2b: Predicted vs Actual Categories (ReLU)

```
Text:
Think!

It's the SCSI card doing the DMA transfers NOT the disks...

The SCSI card can do DMA transfers containing data from any of the SCSI devices
it is attached when it wants to.

An important fea
Actual Category: 6
Predicted Category: 6
--------------------------------------------------
Text: 1)    I have an old Jasmine drive which I cannot use with my new system.
 My understanding is that I have to upsate the driver with a more modern
one in order to gain compatability with system 7.0.1.
Actual Category: 19
Predicted Category: 18
```

Figure 2c: Predicted vs Actual Categories (ReLU)

Leaky ReLU

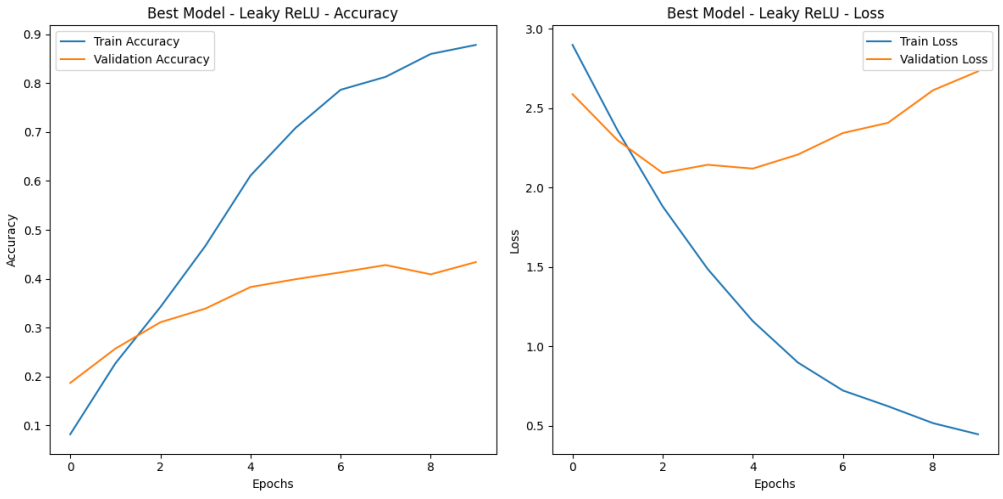

Figure 3: Accuracy and Loss Curves for Leaky ReLU

```
Sample Predictions:
Text:

I am sure some bashers of Pens fans are pretty confused about the lack
of any kind of posts about the recent Pens massacre of the Devils. Actually,
I am  bit puzzled too and a bit relieved. However,
Actual Category: 9
Predicted Category: 9
--------------------------------------------------
Text: My brother is in the market for a high-performance video card that supports
VESA local bus with 1-2MB RAM.  Does anyone have suggestions/ideas on:

  - Diamond Stealth Pro Local Bus

  - Orchid Farenh
Actual Category: 14
Predicted Category: 16
```

Figure 4a: Predicted vs Actual Categories (Leaky ReLU)

```
Text:



        Finally you said what you dream about. Mediterranean???? That was new....
        The area will be "greater" after some years, like your "holocaust" numbers




                *****
        Is't July in USA now?????
Actual Category: 1
Predicted Category: 4
```

Figure 4b: Predicted vs Actual Categories (Leaky ReLU)

```
Text:
Think!

It's the SCSI card doing the DMA transfers NOT the disks...

The SCSI card can do DMA transfers containing data from any of the SCSI devices
it is attached when it wants to.

An important fea
Actual Category: 6
Predicted Category: 0
--------------------------------------------------
Text: 1)    I have an old Jasmine drive which I cannot use with my new system.
 My understanding is that I have to upsate the driver with a more modern
one in order to gain compatability with system 7.0.1.
Actual Category: 19
Predicted Category: 18
```

Figure 4c: Predicted vs Actual Categories (Leaky ReLU)

Swish

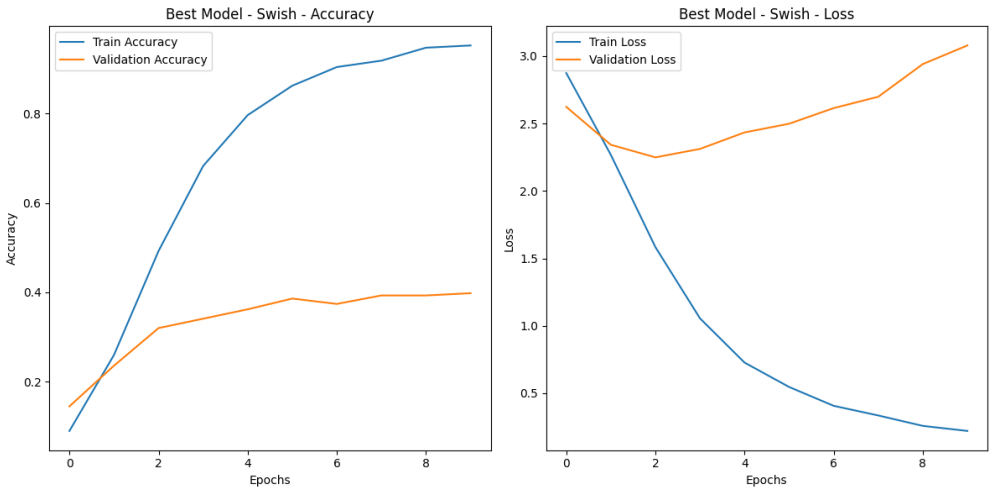

Figure 5: Accuracy and Loss Curves (Swish)

```
Sample Predictions:
Text:

I am sure some bashers of Pens fans are pretty confused about the lack
of any kind of posts about the recent Pens massacre of the Devils. Actually,
I am  bit puzzled too and a bit relieved. However,
Actual Category: 9
Predicted Category: 9
--------------------------------------------------
Text: My brother is in the market for a high-performance video card that supports
VESA local bus with 1-2MB RAM.  Does anyone have suggestions/ideas on:

  - Diamond Stealth Pro Local Bus

  - Orchid Farenh
Actual Category: 14
Predicted Category: 14
--------------------------------------------------
```

Figure 6a: Predicted vs Actual Categories (Swish)

```
Text:



       Finally you said what you dream about. Mediterranean???? That was new....
       The area will be "greater" after some years, like your "holocaust" numbers.




                *****
       Is't July in USA now?????
Actual Category: 1
Predicted Category: 12
```

Figure 6b: Predicted vs Actual Categories (Swish)

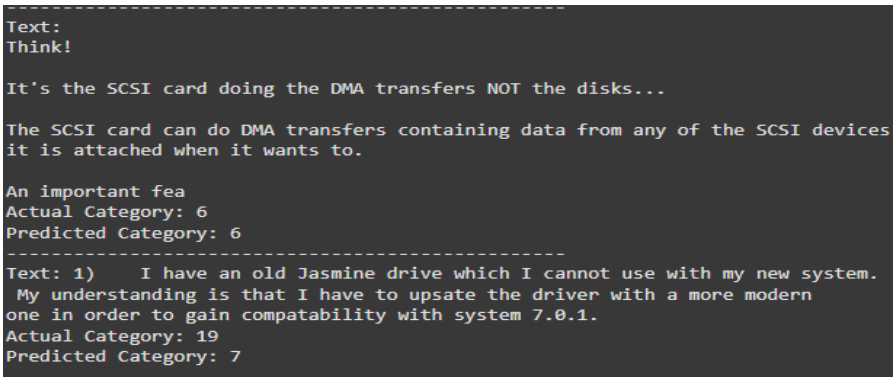

Figure 6c: Predicted vs Actual Categories (Swish)

The performance of activation functions was evaluated ReLU, Leaky ReLU and Swish across the three model configurations in case of text processing (Shallow, Medium, Deep). The comparison focused on their behavior in terms of testing accuracy, loss and training process, observed via the plots and sample output generated (as images).

1. **Performance Comparison**
   a. ReLU
      - The Medium configuration had the highest accuracy among the three configurations, with a test accuracy of 42.5% and test loss of 2.65
      - The Deep configuration had the worst performance with only 26.5% and an extremely high test loss of 3.07
      - While the Shallow Configuration performed relatively mediocre as well 31.2% accuracy and a test loss of 2.47

   b. Leaky ReLU
      - The Medium configuration had again the highest accuracy with a test accuracy of 43.4% and test loss of 2.73
      - The Shallow configuration had the worst performance with test accuracy of only 30.4% and a test loss of 2.38
      - Lastly, the deep configuration performed ok with accuracy of 32.6% but the test loss was high with 3.04

   c. Swish
      - The Medium configuration had the highest accuracy of 39.8% with test loss of 3.08
      - Shallow configuration had the worst performance with lowest accuracy of 32.8% and a test loss of 2.42
      - Lastly, the deep configuration performed decently with accuracy of 34.2% and test loss of 2.69

2. **Plots**:
   a. ReLU:
      - The training accuracy of the model increases in a rapid manner and reaches upto around 88% during the 8th epoch, showing an effective learning on the training set of news data.

- While the validation accuracy as shown in Figure 1, it continues to improves but stabilizes around 45%, showcasing the generalization ability of the model is less effective when compare with its training performance
- The training loss curves show that the loss continues to decrease in a steady manner showing that the optimization was effective. While the validation loss starts to increase after the third epoch which means the model is overfitting

b. Leaky ReLU:
    - The training accuracy improves in a steady fashion, and continues till approximately 90%, showing that the mode is learning well on the training data.
    - The validation accuracy also continues to increase around 45% but then model has difficulty in generalizing after a certain point over the new unseen data. The significant disparity between both accuracy curves are strongly indicative of the model overfitting
    - As figure 3 shows the training loss diminishes after start but eventually reaches near zero showcasing effective and efficient learning on the training data. Meanwhile validation curve increases after the 3rd epoch, again indicative of overfitting

c. Swish
    - The training accuracy continues to improve quite rapidly, reaching near to 90% during the 8th epoch as per Figure 5
    - Meanwhile the validation accuracy, shows slightly better generalization compared to other two function being stable around 45 – 50 %.
    - The training loss reaches a very low value by decreasing consistently showcasing an effective learning behavior. While the validation loss is more indicative that swish generalizes better on the new unseen textual data.

If we analyze all the three functions together Swish's plots show that it generalizes better compared to both RELU and Leaky ReLU, the smoother gradient may allot the model to be able to capture more relationships, both linear and non linear, in the dataset, which might lead to slightly better accuracy percentage and overall lower overfitting. However, overfitting has been observed in all three activation functions.

Predicted vs Actual Outputs:

The code generated 5 sample outputs to see how the best configuration for each activation function performs and is able to classify the data in the right categories. (In total there are 20 categories, the text is generated and classified into a category by a model). The ReLU model was out of 5 sample texts generated and was able to correctly classify 3 out of 5 texts into the right categories. (Correctly classified were 9, 14, 6). While the Leaky ReLU was able to correctly classify only 1 out of 5 texts into the right category. (Correctly classified were 9 only). Lastly swish was able to classify 3 out of 5 texts into the right categories (Categories were 9, 14, 6)

Image Classification

Activation Function	Shallow	Medium	Deep	Deeper
ReLU	59.89%	75.33%	76.54%	74.81%
Leaky ReLU	53.58%	68.84%	77.29%	68.5%
Swish	60.03%	75.61%	79.39%	62.46%

Table 2

ReLU

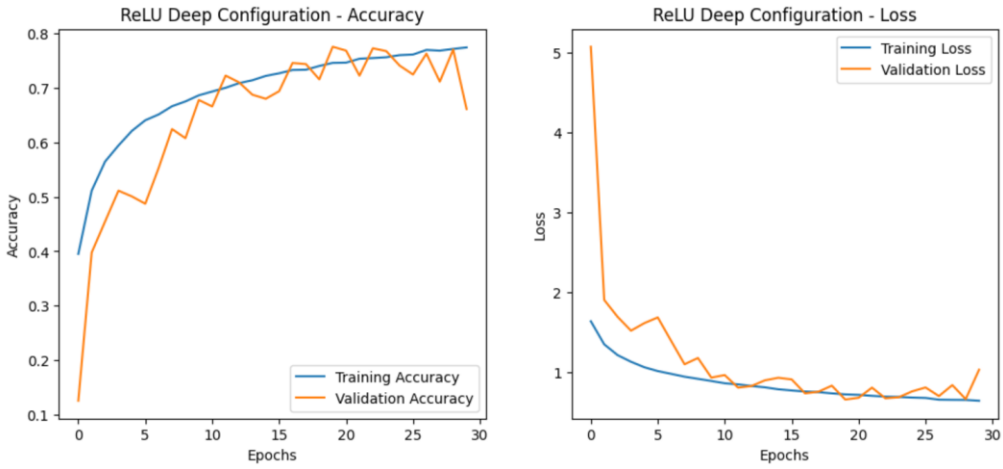

Figure 7: Accuracy and Loss Curves for Relu

Figure 8: Accuracy and Loss Curves classes Relu

Leaky ReLU

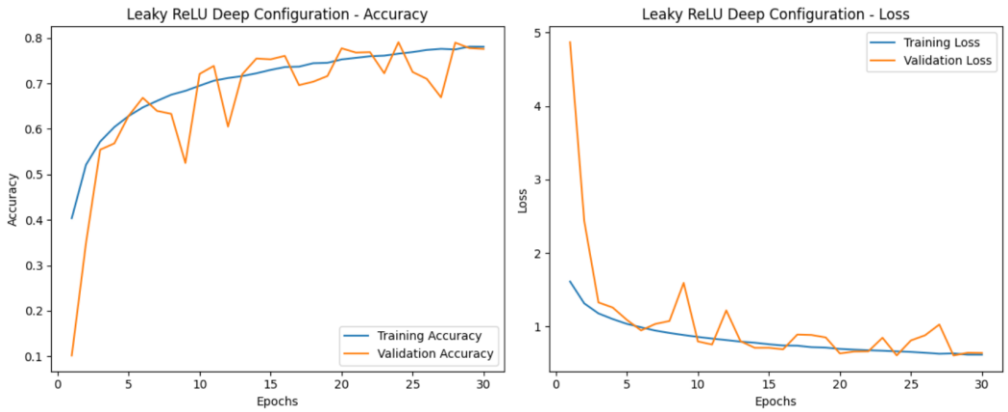

Figure 9: Accuracy and Loss Curves (Leaky Relu)

Figure 10: Predicted vs Actual for classes Leaky Relu

Swish

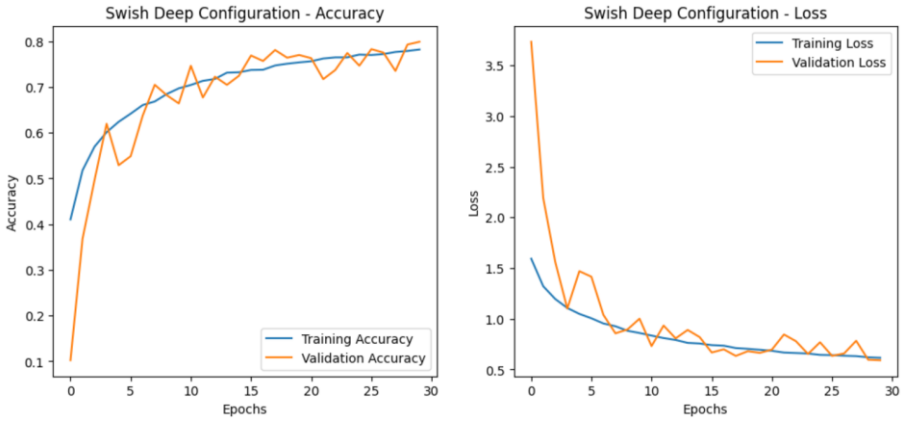

Figure 11: Accuracy and Loss Curves (Swish)

Figure 12: Predicted vs Actual for classes Swish

The performance of the activation functions: ReLU, Leaky ReLU, and Swish, was evaluated across the four model configurations (Shallow, Medium, Deep, and Deeper). The comparison highlights their behavior in terms of test accuracy, test loss, training dynamics observed from the plots, and predictions on sample test images.

1. **Performance Comparison**:

ReLU:

ReLU demonstrated strong performance in most configurations but showed limitations in deeper networks.

- **Deep Configuration:** ReLU achieved its best performance in the deep configuration with a test accuracy of 76.54% and a test loss of 0.6650. This result highlights ReLU's ability to work effectively in moderately deep architectures.
- **Deeper Configuration:** The accuracy dropped to 74.81%, with an increase in test loss to 0.7218. This decline indicates signs of overfitting, which was also reflected in the training and validation curves, where the validation loss began to diverge after a few epochs.
- **Shallow and Medium Configurations:** ReLU performed decently in simpler networks, achieving 55.88% and 74.13% accuracy, respectively. These results suggest that ReLU works well as a baseline activation function but is more sensitive to increasing network complexity, particularly without strong regularization.

While ReLU excels in moderately deep networks, its inability to handle negative gradients and the dying ReLU problem limits its performance in very deep architectures.

Leaky ReLU:

ReLU demonstrated strong performance in most configurations but showed limitations in deeper networks.

- **Deep Configuration:** ReLU achieved its best performance in the deep configuration with a test accuracy of 76.54% and a test loss of 0.6650. This result highlights ReLU's ability to work effectively in moderately deep architectures.
- **Deeper Configuration:** The accuracy dropped to 74.81%, with an increase in test loss to 0.7218. This decline indicates overfitting, which was also reflected in the training and validation curves, where the validation loss began to diverge after a few epochs.
- **Shallow and Medium Configurations:** ReLU performed decently in simpler networks, achieving 55.88% and 74.13% accuracy, respectively. These results suggest that ReLU works well as a baseline activation function but is more sensitive to increasing network complexity, particularly without strong regularization.

While ReLU excels in moderately deep networks, its inability to handle negative gradients and the dying ReLU problem limits its performance in very deep architectures.

Swish:

Swish emerged as the best-performing activation function across the models, particularly in deep architectures.

- **Deep Configuration:** Swish achieved the highest overall test accuracy of 79.39% and the lowest test loss of 0.6058, outperforming both ReLU and Leaky ReLU. The smooth and non-monotonic nature of Swish allowed for better gradient flow, which enhanced learning dynamics and generalization.
- **Medium Configuration:** Swish also performed well here, achieving 75.61% accuracy, slightly higher than ReLU and significantly better than Leaky ReLU.
- **Deeper Configuration:** In contrast to its strong performance in other setups, Swish underperformed significantly in the deeper configuration, achieving only 62.46% accuracy and a test loss of 1.1581. This result indicates that Swish, like other activation functions, is also susceptible to overfitting in extremely deep models without adequate regularization.
- Swish showed smoother accuracy and loss curves across most configurations, with fewer spikes compared to ReLU and Leaky ReLU. This stability indicates that Swish effectively handles gradient flow, particularly in deep networks.

Swish outperformed both ReLU and Leaky ReLU in the deep configuration, owing to its smooth gradient flow and non-monotonic behavior. However, its performance declined in very deep networks, possibly due to overfitting.

2. **Plots**:

The plots of training accuracy, validation accuracy, training loss, and validation loss for the deep configuration of all three activation functions (ReLU, Leaky ReLU, and Swish) provide key insights into their training dynamics and generalization performance.

- **ReLU:**

The accuracy and loss curves for the ReLU activation function demonstrate stable training accuracy but noticeable fluctuations in the validation loss.

- **Training Accuracy and Validation Accuracy:**
  - The training accuracy steadily increases over the epochs, reaching approximately 78% by epoch 30.
  - Validation accuracy lags behind the training accuracy, peaking around 74-75% before dropping slightly in the final epochs. This discrepancy indicates potential overfitting as the model learns the training data well but struggles to generalize to unseen data.
- **Training Loss and Validation Loss:**
  - Training loss decreases smoothly and converges around 0.6.
  - Validation loss initially decreases but shows significant fluctuations (spikes) from epoch 10 onward. These spikes suggest that the model is overfitting, as it fails to improve validation performance consistently.

While ReLU achieves good training accuracy, its inability to propagate gradients for negative inputs (dying ReLU problem) may limit learning dynamics, especially in deeper

layers. The spikes in validation loss confirm that ReLU struggles to generalize in deeper architectures, necessitating stronger regularization.

- **Leaky ReLU:**

  The Leaky ReLU curves exhibit significant variations, particularly in the early epochs, before stabilizing towards the end of training.

  - **Training Accuracy and Validation Accuracy:**
    - The training accuracy increases steadily, reaching about 78%, similar to ReLU.
    - The validation accuracy curve fluctuates heavily during the initial epochs, particularly between epochs 5 and 15. This suggests that Leaky ReLU is more sensitive to the learning rate and weight initialization but eventually stabilizes around 77-78%.
  - **Training Loss and Validation Loss:**
    - Training loss decreases consistently, reaching similar values as ReLU.
    - Validation loss shows more pronounced spikes during early epochs but stabilizes in the later epochs, converging closer to training loss. However, occasional minor spikes indicate that the model is still struggling to generalize perfectly.

  Leaky ReLU outperforms ReLU in terms of validation accuracy and stability over time but remains sensitive during the early training phases. This sensitivity might be due to its small negative slope, which allows gradients to flow for negative inputs but requires careful tuning of learning rates.

- **Swish:**

  The Swish activation function exhibits the most consistent and stable training behavior across both accuracy and loss curves.

  - **Training Accuracy and Validation Accuracy:**
    - Training accuracy steadily increases to about 79% by epoch 30, showing strong convergence.
    - Validation accuracy closely follows training accuracy, with minimal divergence, peaking at around 79%. Unlike ReLU and Leaky ReLU, Swish achieves smoother improvements with fewer spikes, indicating better generalization to unseen data.
  - **Training Loss and Validation Loss:**
    - Training loss decreases smoothly to approximately 0.5, which is lower than both ReLU and Leaky ReLU.
    - Validation loss decreases consistently and remains close to training loss throughout the epochs, with minimal spikes or instability. This suggests

that Swish maintains smooth gradient flow and reduces the risk of overfitting.

Swish demonstrates superior training dynamics compared to ReLU and Leaky ReLU. Its smooth and non-monotonic nature allows for better gradient flow, particularly in deeper layers, leading to improved generalization and reduced validation loss.

3. **Predicted vs Actual Outputs**:

   Sample predictions were displayed for 10 test images, with their true labels and corresponding predicted labels.

   - **ReLU:**
     - Correctly predicts simpler images (e.g., ships and cats).
     - Misclassifies complex images, such as a frog (true: 6) being predicted as a dog (pred: 9).
     - Errors often occur when the features of two classes overlap, suggesting that ReLU may struggle to capture subtle differences in deeper layers.
   - **Leaky ReLU:**
     - Shows improved predictions for certain images compared to ReLU, particularly for simpler classes like cars and ships.
     - However, misclassifications still occur, particularly in complex images, where some frogs are predicted as dogs or other animals.
   - **Swish:**
     - Produces the most accurate predictions, correctly classifying most test images, including challenging cases like animals and vehicles.
     - Minimal errors demonstrate Swish's ability to learn subtle and high-level features effectively, particularly in deeper networks.

   Overall, the **Swish activation function** outperformed ReLU and Leaky ReLU, especially in deeper architectures, due to its ability to smooth gradients and improve training dynamics. However, deeper configurations of all models showed susceptibility to overfitting, which can be further mitigated with additional regularization or larger datasets. The combination of plots, accuracy/loss values, and sample outputs provides a comprehensive analysis of model behavior and activation function effectiveness.

Recommendations

Text Classification

The performance of three functions can be tested on a complete dataset instead of a subset of dataset for better understanding of the performance and generalization ability of all three functions. Furthermore we only mainly tuned the Model Hyperparamters to see the impact those make on the performance of activation functions due limitation of computational resources and time, there is a great scope to actually

understand by tuning different kinds of Parameters some of them are Regularization, Data related, Training, Experimental and Evaluation parameters that can be tinkered with to assess the whether Swish performs better than RelU and Leaky ReLU. There is a need to test it on more deep complex architectures to see validate the Author's claim.

Image Classification

Swish should be used for tasks involving deep architectures or applications where high accuracy and generalization are critical. It is particularly well-suited for complex tasks such as deep image classification, object detection, and other deep learning problems where gradient flow stability is essential. However, its computational cost should be carefully considered for large-scale datasets. ReLU remains the most practical and efficient choice for shallow to moderately deep networks. It is ideal for real-time tasks or applications with limited computational resources, where simplicity and speed are essential. Techniques such as Batch Normalization and Dropout can be applied to improve its performance in moderately deep networks. Leaky ReLU provides an intermediate solution when improved stability over ReLU is needed, but computational efficiency remains a priority. It is particularly suitable for moderately deep networks or scenarios where the dying ReLU problem is a concern.

Future work can include exploring larger datasets (e.g., CIFAR-100) and additional regularization techniques to improve generalization in deeper configurations. Additionally, hardware constraints should be considered, as Swish may demand more computational resources compared to ReLU and Leaky ReLU.

Conclusion

In our project we conducted a thorough research and developed two models to assess the performance of Swish, ReLU and Leaky Rule on text and images data. As per our results Swift function performed the best in configurations we tested on (three for text) and (four for images). However, consistency is still a question mark which is to answered whether the performance gains of Swish activation function outweigh the additional resources and cost its use incurs in complex deep learning architectures.

For the both the text and image data, the experiment clearly showed that the choice of activation function plays a critical role in determining the performance and generalization capabilities of GRU's and CNN's especially as network complexity increases. Among the activation functions tested, Swish emerged as the top performer, particularly in deeper networks. Its smooth and non-monotonic properties enabled better gradient flow, more stable training dynamics, and improved generalization. However, the computational cost of Swish makes it less practical for resource-limited environments.

ReLU proved to be a reliable baseline for moderate (text) and shallow to moderately deep networks (images) due to its simplicity, computational efficiency, and solid performance. Nevertheless, in deeper architectures, it showed a tendency to overfit, largely due to the "dying ReLU" problem where gradients become zero for certain inputs. Leaky ReLU helped address this limitation by allowing gradients to flow even for negative inputs, offering more stability compared to ReLU. However, its performance improvements were relatively minor when compared to Swish.

Based on these findings, Swish is recommended for tasks that involve deeper networks and demand high accuracy, while ReLU and Leaky ReLU remain strong, efficient alternatives for shallower or moderately deep networks. Future research can explore larger datasets, incorporate advanced regularization techniques, and focus on further optimizing activation functions to enhance performance in extremely deep neural architectures.

References

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv preprint arXiv:1502.01852*.

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289*.

Banerjee, C., Mukherjee, T., & Pasiliao, E. (2020). The Multi-phase ReLU Activation Function. *Proceedings of the 2020 ACM Southeast Conference*.

Rane, C., Tyagi, K., Kline, A., Chugh, T., & Manry, M. (2024). Optimizing Performance of Feedforward and Convolutional Neural Networks through Dynamic Activation Functions. *Evolutionary Intelligence*

Lakhdari, K., & Saeed, N. (2022). A new vision of a simple 1D Convolutional Neural Networks (1D-CNN) with Leaky-ReLU function for ECG abnormalities classification. Intelligence-Based Medicine, 6, 100080.

Mastromichalakis, S. (2020). ALReLU: A different approach on Leaky ReLU activation function to improve Neural Networks Performance. arXiv preprint arXiv:2012.07564.

Maniatopoulos, A., & Mitianoudis, N. (2021). Learnable Leaky ReLU (LeLeLU): An Alternative Accuracy-Optimized Activation Function. Information, 12(12), 513.

Lamprou, Z., Tenedios, I., & Moshfeghi, Y. (2024). On the Role of Activation Functions in EEG-To-Text Decoder. arXiv preprint arXiv:2410.12572.

Subramanian, S., et al. (2018). Towards Text Generation with Adversarially Learned Neural Outlines. NeurIPS 2018.

Chieng, H. H., Wahid, N., Ong, P., & Perla, S. R. K. (2018). Flatten-t swish: A thresholded ReLU-swish-like activation function for deep learning. *arXiv Preprint arXiv:1812.06247*.

Eger, S., Youssef, P., & Gurevych, I. (2019). Is it time to swish? Comparing deep learning activation functions across NLP tasks. *arXiv Preprint arXiv:1901.02671*.

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. *Proc. Icml*, *30*, 3.

Mercioni, M. A., & Holban, S. (2020). P-swish: Activation function with learnable parameters based on swish activation function in deep learning. *2020 International Symposium on Electronics and Telecommunications (ISETC)*, 1–4.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814.

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv Preprint arXiv:1710.05941*.