**Training Multi Layer Perceptron with different Activation Functions**

**Expected Observations and Results**

1. ReLU (Rectified Linear Unit):

* Generally performs the best for this type of image classification task
* Faster convergence due to no vanishing gradient problems for positive values
* Sparse activation (some neurons output exactly zero) which can help with feature selection
* May suffer from "dying ReLU" problem where neurons can get stuck in a permanently inactive state

1. LeakyReLU:

* Usually performs similarly to ReLU but can be slightly more robust
* Helps prevent the "dying ReLU" problem by allowing small negative gradients
* May converge slightly slower than ReLU but often reaches similar final accuracy
* Can be more stable during training due to the small negative slope

1. Sigmoid:

* Typically performs worse than ReLU and LeakyReLU on deep networks
* Slower convergence due to vanishing gradient problems
* Outputs are bounded between 0 and 1, which can cause saturation
* Historical activation function but generally not recommended for hidden layers in modern deep learning

Expected patterns in the results:

1. Training Speed: ReLU and LeakyReLU should train faster than Sigmoid
2. Final Accuracy: ReLU and LeakyReLU should achieve similar final accuracies, while Sigmoid likely lags behind
3. Learning Stability: LeakyReLU might show slightly more stable learning curves than ReLU
4. Convergence: Sigmoid might struggle to reach the same level of accuracy even with extended training

**Actual Observations and Results**

**A graph of different colored lines

Description automatically generated**

1. Overall Performance Comparison:

* LeakyReLU performed best with 49.07% test accuracy
* ReLU came in second with 47.11% test accuracy
* Sigmoid performed worst with 46.40% test accuracy

1. Training Dynamics Analysis:

Training Accuracy (Left Plot):

* All three models show similar learning patterns in the early epochs
* LeakyReLU and ReLU achieved slightly higher final training accuracies (around 51%) compared to Sigmoid (50%)
* The learning curves are relatively smooth, indicating stable training
* All models show continuous improvement throughout training, suggesting they might benefit from additional epochs

Validation Accuracy (Right Plot):

* More volatile than training accuracy, showing typical fluctuations
* LeakyReLU shows better stability in later epochs
* ReLU shows some signs of overfitting with larger fluctuations
* Sigmoid shows the most conservative learning pattern but also the lowest peak performance

1. Key Observations:

LeakyReLU:

* Best overall performance
* Most stable validation accuracy in later epochs
* Smallest gap between training and validation accuracy, suggesting better generalization
* Final test accuracy (49.07%) is notably better than the other activations

ReLU:

* Shows competitive performance but with more validation instability
* Higher variance in validation accuracy suggests some overfitting
* Still outperforms Sigmoid despite the instability

Sigmoid:

* Most conservative learning pattern
* Lowest final performance
* Smaller fluctuations in validation accuracy but also lower peak performance
* Shows signs of saturation in later epochs

1. Recommendations:
2. Model Improvements:

* Consider increasing the number of epochs as all models show potential for further learning
* Add batch normalization layers to help with training stability
* Experiment with dropout layers to reduce overfitting, especially for ReLU
* Try a learning rate schedule to improve convergence

1. Activation Function Choice:

* LeakyReLU appears to be the best choice for this specific architecture and dataset
* If stability is a priority, LeakyReLU would be the recommended choice
* Consider using LeakyReLU with a slightly higher alpha value (>0.01) to potentially improve performance further

1. Architecture Modifications:

* The relatively low accuracy across all activations suggests the model might benefit from additional capacity
* Consider adding more layers or increasing the number of neurons
* Adding convolutional layers might significantly improve performance on this image classification task