**Regularization Strategies on KMNIST Dataset**

**Experiment Setup and Baseline Architecture**

**Objective of the Experiment**

The aim of this experiment is to evaluate the performance of Convolutional Neural Networks on the Kuzushiji-MNIST dataset which contains the handwritten Japanese characters. The primary focus of these experiments is to analyze the different architectural choice and regularization techniques which impact the classification accuracy.

The study investigates:

1. Effectiveness of using the CNN in extracting the features from the KMNIST images.
2. Assess the impact of regularization techniques on generalization.
3. Optimization strategies for improving the convergence and reducing the overfitting.

**Experimental Design**

**The experiment will follow the below approach:**

The baseline CNN architecture has been predefined and will remain unchanged initially.

Hyperparameters learning rate, optimizer setting, batch size has been adjusted to analyze their effects on model performance.

Regularization techniques like dropout, stochastic gradient optimizer with momentum and L2 regularization have been introduced progressively.

Performance of the model is measured using the test accuracy, precision and recall.

**Baseline CNN Architecture**

**The baseline model consists of:**

**Five Convolutional layers**

1. With increasing number of filters: (32, 64, 128, 256, 512)
2. Kernel size: 3x3
3. Activation function: ReLU
4. Maximum pooling applied for some layers which helps in reducing the spatial dimensions.

**Fully Connected Layers:**

1. Flatten layers which convert the feature maps into a vector.
2. First layer: 512 inputs, 512 outputs with ReLU activation.
3. Output layer: 512 inputs with 10 output neurons refers to the multi classes.

**Architecture Justification:**

The architecture will progressively extract the low level to higher level features from small edges to the complex patterns.

Pooling will reduce the spatial dimensions thereby preventing the overfitting and will reduce the computational cost.

Fully connected layers will allow the extracted features to contribute to the classification.

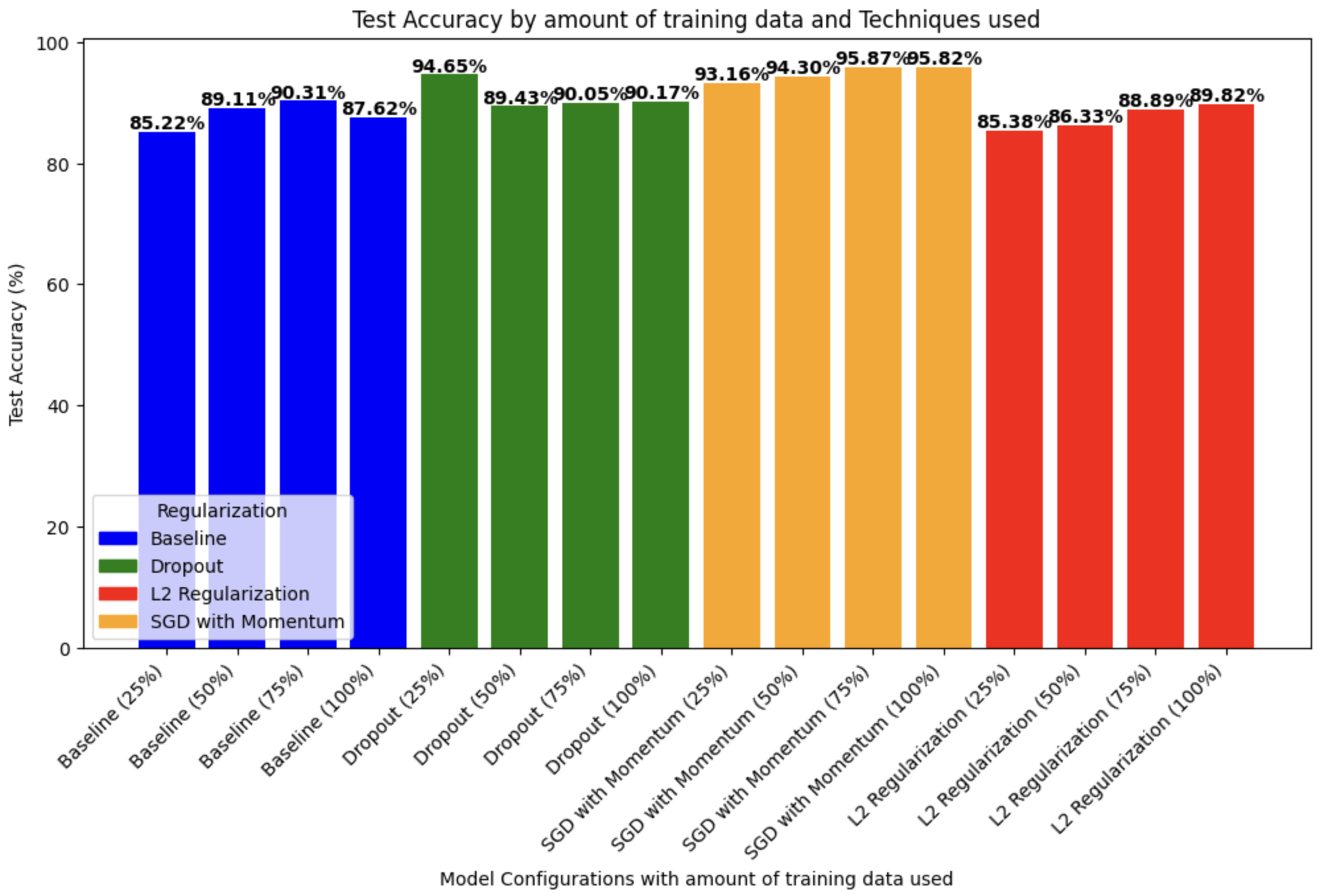
**Training Setup and Hyperparameters**

| **Hyperparameter** | **Value** |
| --- | --- |
| Dataset | KMNIST (28x28 grayscale images) |
| Optimizer | Stochastic gradient descent |
| Learning rate | 0.01 |
| Loss function | Cross entropy loss |
| Batch size | 64 |
| Epochs | 5 |
| Weight Initialization | Default PyTorch Initialization |
| Regularization | None for baseline |
| Hardware used | GPU if available, else CPU. |
| Momentum | 0.9 |
| Weight Decay | 0.0001 (L2 Regularization) |
| Dropout | 20% |

**Hypothesis**

We expect the baseline CNN Architecture will reach the classification accuracy of ~90% but it may show overfitting due to lack of regularization. In further experiments, which introduces the dropout, stochastic gradient optimizer with momentum and L2 regularization should enhance the generalization thereby improving the test accuracy.

**Results**

****

*Figure1: Showcases the test accuracy for the baseline and regularization techniques applied to architectures with the amount of training data used.*

|  | **Metric** | **100% Training data** | **25% Training data** | **50% Training data** | **75% Training data** |
| --- | --- | --- | --- | --- | --- |
| **Baseline** | **Precision** | 0.89 | 0.87 | 0.90 | 0.91 |
|  | **Recall** | 0.88 | 0.85 | 0.89 | 0.90 |
| **Dropout** | **Precision** | 0.90 | 0.95 | 0.90 | 0.90 |
|  | **Recall** | 0.90 | 0.95 | 0.89 | 0.90 |
| **SGD with Momentum** | **Precision** | 0.96 | 0.93 | 0.94 | 0.96 |
|  | **Recall** | 0.96 | 0.93 | 0.94 | 0.96 |
| **L2 Regularization** | **Precision** | 0.90 | 0.87 | 0.88 | 0.90 |
|  | **Recall** | 0.90 | 0.85 | 0.86 | 0.89 |

*Table 1: showcases the precision and recall for the baseline and regularization techniques applied to architectures with the amount of training data used*

**Analysis: Insights from Experiments**

**Trend 1: Regularization Improves accuracy, precision and recall.**

SGD with momentum is the best performer with both the highest test accuracy (93 to 94)% and best precision 0.96 and the recall 0.96. Higher precision means the fewer false positives which indicates SGD with Momentum will refine the updates more efficiently.

SGD with momentum has benefited from the additional data where the test accuracy has improved as the size of the data increased which tells us that momentum based updates stabilize when the training is done on larger datasets.

Dropout improved the data efficiency helping the model to generalize well with less training data where 94.65% test accuracy is achieved with 25% of training data. Alongside has the 0.95 recall, indicates fewer false negatives thereby we can conclude that dropout prevents the overfitting which leads to better generalization.

L2 regularization has provided the stable improvements but slightly weaker in the recall which tells that the model doesn’t generalize well to all the classes.

**Trend 2: The Trade-Off Between the Training Data and Regularization Efficiency.**

Dropout performs better with the less data compared to the other techniques, with 0.95 recall and precision at 25% of training data, while the baseline model has dropped significantly in performance which indicates diverse feature learning even with the less data.

SGD with momentum will provide the highest accuracy and stability compared to the other techniques but requires 100% of training data.

L2 Regularized has struggled with the limited data, which has shown larger drops in precision, recall and accuracy when the trained data is reduced.

The results confirmed the hypothesis that regularization techniques have improved the model generalization, with SGD with momentum achieving the highest accuracy, precision and recall but it required more training data. Dropout is the most data-efficient method which has maintained the storing performance even at 25% of the training data which has outperformed the baseline at 100% training data. This concludes that the regularization has not only prevented the overfitting but also reduced the dependency on a large dataset thereby making the model more robust and efficient.