1. Introduction

We are doing a classification task by trying to predict which mathematical symbol is drawn on an image using the HASYv2 dataset with 20 different classes only. We will use 2 types of deep networks for this task: a Multilayer Perceptron and a Convolutional Neural Network, and a dimensionality reduction technique: the Principal Component Analysis.

The dataset used in this project exhibits a relatively balanced distribution among its classes, so accuracy can be a useful performance metric. However, to ensure that our evaluation is not biased towards the most represented class, we will also assess the models using the F1-score. By considering accuracy and F1-score, we can obtain a more comprehensive evaluation of the models' performance on the dataset.

2. Methods

- **2.1 Data preparation:**

Data normalization: Z-score normalization is used to reduce the effect of differences in feature scales on the outcome. This led to better model convergence and faster training times.

- **2.2 Cross Validation**

We are using a 1-Fold Cross Validation method in which the raw data is separated as follows: 80% is used for training and 20% for testing. The training set is split as follows: 20% for validation and the rest is for fitting the model.

- **2.3 Deep Networks**

We conducted a systematic search using a random search technique to find the best hyperparameters and architecture for each model.

- **Multilayer Perceptron (MLP)**

In the first time, we aimed to determine the optimal architecture by varying the number of hidden layers and neurons, we tested 3 different ones. We kept the learning rate fixed to 1e-1, used the ReLU activation function and set the number of epochs to 100. In the second time, due to computational limitations, we performed cross-validation on both the learning rate and activation functions. Moreover, we used Stochastic Gradient Descent as the optimizer and Cross-entropy as the loss function.

Architecture	1	2	3
nb of hidden layer(s) /nb of nodes per hidden layer(s)	1/ 1500	2/ 2500	3/ 2000
Accuracy (%)	91.2	91.0	88.1
F1-Score	90.2	89.8	87.2

Figure 1: Accuracies on train set for different MLP architectures with lr=1e-1 and ReLU

To find the best hyperparameters for MLP, we evaluated the model with a range of learning rates [1e-1, 1e-2, 1e-3, 1e-4] and 2 different activation functions, ReLu and Sigmoid. In conclusion, architecture 1 gives the best accuracy = **86.4%** on the validation set with a learning rate of 1e-1 and the ReLU function. It has 1,567,520 trainable parameters which is much lower than the other two.

- **Convolutional Neural Network (CNN)**

CNN has more parameters than MLP. We applied the same methodology as before with 3 different architectures (with different numbers of convolutional layers and fully connected layers) and we used cross-validation. Similarly, the loss function and optimizer are the same. All architectures have these hyperparameters : kernel_size=3, padding=1, stride=1, and max pooling.

Architecture	1	2	3
Convolution layers with filter sizes	[32,64]	[64, 128, 128,128]	[64, 128, 256]
Nodes per hidden layer	128	64 & 128	64
Accuracy	93.2%	92.3%	94.5%
F1-Score	92.2%	91.1%	93.8%

Figure 2: Architectures for the CNN and their respective accuracy & F1-score on the train set (lr=1e-1, activation function=ReLU)

On the validation set, our best architecture is the n°3 with a learning rate of 1e-2 and the ReLu activation function with an accuracy of **94.5%**. We have a total of 903,636 trainable parameters.

- **2.4 Principal Component Analysis (PCA)**

PCA is a process that reduces the dimensionality of the dataset. This is why we used it with other methods to improve our runtime. Moreover, PCA is unsupervised and thus may not always preserve category information. This means that the accuracy may be lower with PCA enabled than without. We have reduced all the methods to the same dimension d=200, which has an explained variance of 83.39%. This means that we have obtained a good tradeoff between dimensionality reduction and retaining an acceptable level of information.

3. Experiment/Results

The cross-validation for CNN and MLP allows us to remark that the sigmoid function decreases our accuracy for all the cases. It is certainly due to the vanishing gradient problem which can be solved using a Res-Net architecture.

Model	Accuracy	F1-score
MLP	86.4%	85.9%
CNN	94.5%	93.7%

Figure 4: Accuracies on test set with the best architectures

Consistently, the CNN exhibited the highest test performance as anticipated, achieving an accuracy of 94.5%. This outcome aligns with expectations due to CNN's specialized design for image processing tasks.

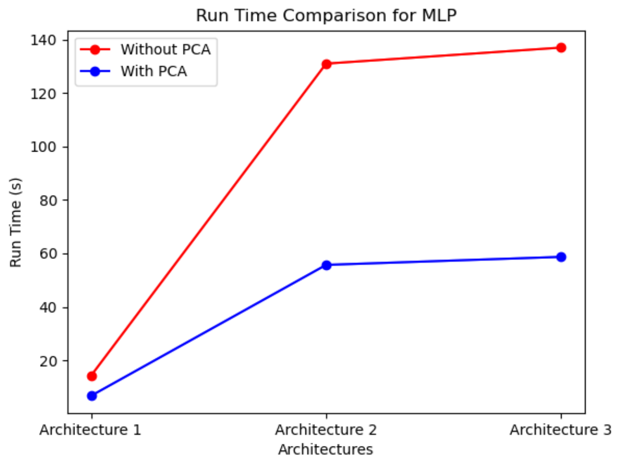

Figure 3: this figure shows an important performance gain of a factor 2 by using PCA.

Regarding PCA, we have tested the performance and accuracy differences of Kmeans and MLP. We thus confirm that the accuracies don't change, meaning that our tradeoff was well chosen and that images lie in small dimensions. Regarding the runtime, we see a two fold diminution on the MLP method (shown in Figure 3) and a four fold diminution on the Kmeans, using k=35.

In conclusion, for further enhancement of our analysis, an alternative approach would involve employing LDA (Linear Discriminant Analysis) instead of PCA. LDA is particularly advantageous as it effectively addresses classification tasks by managing both between-class and within-class variances.