# Assignment 3: Classification of Image Data

COMP 551 Winter 2024, McGill University
Contact TAs: Safa Alver (safa.alver@mcgill.ca)

Released on February 28
Due on April 1 midnight

**Please read this entire document before beginning the assignment.**

## Preamble

- This assignment is **due on April 1st at 11:59pm (EST, Montreal Time).**
- For late submission, $2^k$ percent will be deducted per $k$ days of the delay.
- To use your 6-day quota as a team, submit your request by emailing `comp551.socs@mcgill.ca` with subject title "A3 extension request" and in the email body specify the number of days (max at 6 days) you need to submit your assignment. You can only submit the request ONCE. Once you request the days, the quota of every member on the team will be reduced by the days you requested even if you end up submitting your assignment prior to the extended deadline. Therefore, plan and use your quota wisely.
- This assignment is to be completed in groups of three. All members of a group will receive the same grade except when a group member is not responding or contributing to the assignment. If this is the case and there are major conflicts, please reach out to the contact TA or instructor for help and flag this in the submitted report. Please note that it is not expected that all team members will contribute equally. However every team member should make integral contributions to the assignment, be aware of the content of the submission and learn the full solution submitted.
- You will submit your assignment on MyCourses as a group. You must register your group on MyCourses and any group member can submit. See MyCourses or here for details.
- We recommend to use **Overleaf** for writing your report and **Google colab** for coding and running the experiments. The latter also gives access to the required computational resources. Both platforms enable remote collaborations.

- You should use Python for this and all assignments. You are free to use libraries with general utilities, such as matplotlib, numpy and scipy for Python, unless stated otherwise in the description of the task. In particular, in most cases you should implement the models and evaluation functions yourself, which means you should not use pre-existing implementations of the algorithms or functions as found in SciKit learn, and other packages. The description will specify this in a per case basis.

## Synopsis

In this miniproject, you will **implement a multilayer perceptron from scratch**, and use it to **classify image data**. The goal is to implement a basic neural network and its training algorithm from scratch and get hands-on experience with important decisions that you have to make while training these models. You will also have a chance to experiment with **convolutional neural networks**.

## Task 1: Acquire the data

Your first task is to acquire the image dataset. You will be using only one dataset in your experiments: **Sign Language MNIST**. Use this dataset with the default train and test partitions. You can use existing machine learning libraries to load the dataset. Note that while working with multilayer perceptrons, after loading the data, you will have to vectorize it so that it can have the appropriate dimensions. Also do not forget to normalize the training and test set (see `https://cs231n.github.io/neural-networks-2/#datapre`). The vectorized pixels are the input features to the classifier. You may also use the tutorial of `https://www.kaggle.com/code/sayakdasgupta/sign-language-classification-cnn-99-40-accuracy` for preprocessing the dataset.

## Task 2: Implement an MLP to classify image data

As covered in Module 5, an MLP is composed of three types of layers: (1) an input layer, (2) hidden layers, (3) an output layer (see Figure 1). Your implementation should include the backpropagation and the mini-batch gradient descent algorithm used (e.g., Stochastic Gradient Descent as covered in Module 4.4).

You should follow the equations that are presented in the lecture slides, and you must implement it from scratch (i.e., you **cannot** use Sklearn, TensorFlow, PyTorch, or any other existing library). Using the Numpy package is encouraged. You are allowed to make use or adapt the Colab code provided from the course Github website (`https://colab.research.google.com/github/yueliyl/comp551-notebooks/blob/master/MLP.ipynb` or `https://colab.research.`

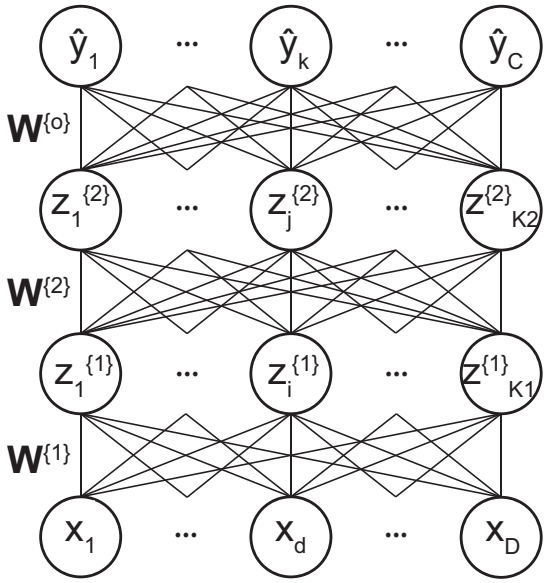


Figure 1: A 2-layer MLP with $D$ input units, $2$ hidden layers with $K1$ and $K2$ hidden units, respectively, and an output layer with $C$ output units.

google.com/github/yueliyl/comp551-notebooks/blob/master/NumpyDeepMLP.ipynb) (but **not** the colab code that uses Tensorflow and Pytorch).

Regarding the implementation, we recommend the following approach:

- Implement the MLP as a Python class. The constructor for the class should take as input the activation function (e.g., ReLU), the number of hidden layers (e.g., $2$) and the number of units in the hidden layers (e.g., $[64, 64]$) and it should initialize the weights and biases (with an initializer of your choice) as well as other important properties of the MLP.
- The class should have (at least) two functions:
  - A `fit` function, which takes the training data (i.e., $\mathbf{X}$ and $\mathbf{y}$)—as well as other hyper-parameters (e.g., the learning rate and number of gradient descent iterations)—as input. This function should train your model by modifying the model parameters.
  - A `predict` function, which takes a set of input points (i.e., $\mathbf{X}$) as input and outputs predictions (i.e., $\hat{\mathbf{y}}$) for these points.
- In addition to the model classes, you should also define a function `evaluate_acc` to evaluate the model accuracy. This function should take the true labels (i.e., $\mathbf{y}$), and target labels (i.e., $\hat{\mathbf{y}}$) as input, and it should output the accuracy score. Feel free to reuse the one you implemented in the previous assignments.

To ensure your MLP implementation work without bug, you will need to verify the gradients (i.e., the Jacobian matrix) computed to update each layer weights is correct by small perturbation analysis as you did in Assignment 2. Feel free to use the code you already have for A2 or use existing function like this one `https://docs.scipy.org/doc/scipy/reference/generated/`

scipy.optimize.check_grad.html.

You are also free to use any Python libraries you like to tune the hyper-parameters; see for example https://scikit-learn.org/stable/modules/grid_search.html.

## Task 3: Run the experiments and report

The goal of the experiments in this part is to have you explore the consequences of important decisions made while training neural networks. *Use test set to estimate performance in all of the experiments after training the model with training set. Evaluate the performance using accuracy.* You are welcome to perform any experiments and analyses you see fit (e.g., the effect of data augmentation, number of hidden layers, number of hidden units, etc in terms of their impacts on accuracy), **but at a minimum you must complete the following experiments in the order stated below**:

1. Create three different models: (1) an MLP with no hidden layer, i.e., it directly maps the inputs to outputs (*Hint: you may find some of your code for Assignment 2 comes in handy for this one*), (2) an MLP with a single hidden layer having ReLU activations, (3) an MLP with $2$ hidden layers having ReLU activations. For each of these models experiment with using $\{32, 64, 128, 256\}$ hidden units and choose the best one. It should be noted that since we want to perform classification, all of these models should have a softmax layer at the end. After training, compare the test accuracy of these three models on the Sign Language MNIST dataset. Comment on how non-linearity and network depth affects the accuracy. Are the results that you obtain expected?

2. Take the last model above, i.e., the one with $2$ hidden layers, and create two different copies of it in which the activation functions are now sigmoid and Leaky-ReLU. After training these two models compare their test accuracies with model having ReLU activations. Comment on the performances of these models: which one is better and why? Are certain activations better than others? If the results are not as you expected, what could be the reason?

3. Take the last model above, i.e., the one with $2$ hidden layers and add $L2$ regularization ($\lambda||\mathbf{W}^{\{\ell\}}||_2^2$) to the network weights for layers $\ell \in \{1, 2, o\}$ and train the MLP in this way. How does this affect the accuracy? The $\lambda$ can be varied as a tunable hyperparameter that can be explored as part of other project requirements.

4. Using existing libraries such as Keras/TensorFlow or PyTorch, create a Convolutional Neural Network (ConvNet) with $3$ convolutional and $2$ fully connected layers. You are free in your choice of the hyperparameters of the convolutional layers. Experiment again with using $\{32, 64, 128, 256\}$ hidden units in the fully connected layers and choose the best one. Also, set the activations in all of the layers to be ReLU. Train this ConvNet on the Sign Language MNIST dataset. Does using a ConvNet increase/decrease the accuracy com-

pared to using MLPs? Provide comments on your results.

5. Using your MLP implementation, try to come up with an MLP architecture that performs as well as possible. How does this MLP perform compared to the ConvNet in part 4? Justify your choice of network architecture parameters through careful experiments.

6. You can report your findings either in the form of a table or a plot in the write-up. **Include in your colab notebook plots of the testing and training accuracy of the MLPs/ConvNet as a function of training epochs.** This will allow you to see how much the network should be trained before it starts to overfit to the training data.

**Note 1: The above experiments are the minimum requirements that you must complete; however, this project is open-ended.**

For example, you might investigate the effect of the ConvNets' hyperparameters related to their convolutional layers (number of filters, filter size, stride, padding, ...) on its test accuracy. It is also possible to examine the effect of different dropout node proportions on the final performance. Another interesting thing to report might be training the MLP and ConvNet with $10^k, k \in \{0, 1, 2, 3, 4\}$ images and plotting the test accuracy. You do not need to do all of these things or tune every parameter, but you should demonstrate creativity, rigour, and an understanding of the course material in how you run your chosen experiments and how you report on them in your write-up.

**Note 2: We expect you to provide plots/tables in your report that justifies your choice of hyperparameters (the learning rates of the MLPs/ConvNets in parts 1-5, the architectural parameters of the MLPs/ConvNets in parts 1, 4 & 5). You are not required to perform cross-validation in this assignment.**

# Deliverable

You must submit two separate files to MyCourses (**using the exact filenames and file types outlined below**):

1. `assignment3_group-k.ipynb`: Your data processing, classification and evaluation code should be all in one single Jupyter Notebook. Your notebook should reproduce all the results in your reports. The TAs may run your notebook to confirm your reported findings.

2. `assignment3_group-k.pdf`: Your (max 8-page) assignment write-up as a pdf (details below).

where k is your group number.

### Write-up instructions

Your team must submit a project write-up that is a maximum of **8 pages** (single-spaced, 11pt font or larger; minimum 0.5 inch margins, an extra page for references/bibliographical content can be used). We highly recommend that students use LaTeX to complete their write-ups. You have some flexibility in how you report your results, but you must adhere to the following structure and minimum requirements:

**Abstract (100-250 words)** Summarize the project task and your most important findings.

**Introduction (5+ sentences)** Summarize the project task, the datasets, and your most important findings. This should be similar to the abstract but more detailed. You should include background information and a few citations to relevant work (e.g., other papers analyzing these datasets).

**Datasets (5+ sentences)** Very briefly describe the dataset. Present the exploratory analysis you have done to understand the data, e.g. class distribution.

**Results (7+ sentences, possibly with figures or tables)** Describe the results of all the experiments mentioned in **Task 3** (at a minimum) as well as any other interesting results you find (Note: demonstrating figures or tables would be an ideal way to report these results).

**Discussion and Conclusion (5+ sentences)** Summarize the key takeaways from the project and possibly directions for future investigation.

**Statement of Contributions (1-3 sentences)** State the breakdown of the workload across the team members.

## Evaluation

The mini-project is out of 100 points, and the evaluation breakdown is as follows:

- Completeness (20 points)
  - Did you submit all the materials?
  - Did you run all the required experiments?
  - Did you follow the guidelines for the project write-up?
- Correctness (40 points)
  - Are your models implemented correctly?
  - Are your reported accuracies close to our solution?
  - Do you observe the correct trends in the experiments (e.g., how the accuracy changes as the depth of the MLP increases)?

  - Do you observe the correct impact of activation choice, regularization and normalization on the model performance?

- Writing quality (30 points)
  - Is your report clear and free of grammatical errors and typos?
  - Did you go beyond the bare minimum requirements for the write-up (e.g., by including a discussion of related work in the introduction)?
  - Do you effectively present numerical results (e.g., via tables or figures)?
- Originality / creativity (10 points)
  - Did you go beyond the bare minimum requirements for the experiments?
  - **Note:** Simply adding in a random new experiment will not guarantee a high grade on this section! You should be **thoughtful and organized** in your report.

# Final remarks

You are expected to display initiative, creativity, scientific rigour, critical thinking, and good communication skills. You don't need to restrict yourself to the requirements listed above - feel free to go beyond, and explore further. You can discuss methods and technical issues with members of other teams, but **you cannot share any code or data with other teams.**