Project #3 Description

Deep Learning

CSE583/EE552 Pattern Recognition and Machine Learning, Spring 2023

Release Date: Thursday, Feb 21, 2023
Submission Due Date: Monday, March 13, 2023

Contents

Do not wait until the deadline for this project since some steps can take a while to run. Start this project early.

1 Part 1: Taiji Keypose Classification with MLP

1.1 Introduction

In Part 1 of this project, we will build on the Taiji key pose classification task and introduce a new approach using a Multilayer Perceptron (MLP) model for classification.

1.2 Dataset

You will be working with the PSU-TMM100 dataset, also used in the previous Project 2. However, to facilitate training with an MLP, we have also provided a foot pressure down-sampled version (lod4) of the dataset. You can access them both on Canvas.

The Taiji dataset contains (N) 49774 samples and has (D) 1961 features in the full version and (D) 68 features in the lod4 version. The dataset is composed of (C) 46 classes, including 45 specific Taiji movements and an additional class for "NON-KEY FRAME" samples, as Project 2 introduced.

Table 1 shows the distribution of samples for training and testing, using the leave-one-subject-out (LOSO) split strategy. You can verify these statistics by examining the provided dataset.

Subject for Testing	Training Samples (Other Subjects)	Testing Samples (Left-Out Subject)
1	44526	5248
2	47483	2291
3	44813	4961
4	44039	5735
5	45686	4088
6	44070	5704
7	44579	5195
8	43996	5778
9	44852	4922
10	43922	5852

Table 1: Number of Training and Testing Samples for each Left-Out Subject in the Taiji Dataset using the LOSO Split Strategy.

1.3 Implementation

To complete Part 1 of the project, you will need to train and evaluate an Multi-Layer Perceptron (MLP) model on the Taiji dataset. We have provided a basic MLP architecture in the starter code, which you can use as a starting point. However, you will be tasked with modifying the network's architecture (layers, activations, etc.) and seeing the model's performance. You shall try at least two different network architectures and report results on them. **Note**: the models should still be able to run in a reasonable amount of time with a CPU. Also, be sure to include justification for the architectural changes you made and insight into the model's performance.

Before training the model, it is important to adjust the training settings to suit your computer's performance.

Layer	Input Dimension	Output Dimension
Linear	input_dim	hidden_dim
Linear	hidden_dim	hidden_dim
Linear	hidden_dim	output_dim

Table 2: Provided basic MLP architecture.

1.4 Requirements

- Train the baseline model and evaluate its performance on both Taiji datasets ($fp\_size = full, lod4$) for each LOSO iteration.
- Implement two different MLP architectures. Evaluate the performance of your models on both Taiji datasets ($fp\_size = full, lod4$) for each LOSO iteration. Provide an explanation of the variations you implemented, the differences between each other as well as the baseline, and some justification on why you tried the architecture.
- To evaluate the performance of each method AND each subset, you must:
  - Display classification results (confusion matrix, overall accuracy & standard deviation, and the classification accuracy for each class) in either figures or tables for each LOSO iteration.
  - Perform t-SNE visualization on the features learned by the model. You must perform t-SNE visualization on the last fully connected layer of each model for ONE subject iteration. Provide figures to support your findings and describe your observations.

1.5 Grading Criteria of Part 1 (30 Points)

- Train and Test the Baseline Model (**5 points**)
    1. Achieve reasonable performance on both Taiji datasets ($fp\_size = full, lod4$) for each LOSO iteration. (5 points)
- Train and Test at Least Two Models (**10 points**)
    1. Achieve better performance on both Taiji datasets ($fp\_size = full, lod4$) for each LOSO iteration. (5 points)
    2. Code implementation. (5 points)
- Report Part (**15 points**)
    1. Display classification results for each method AND each subset. (5 points)
    2. Perform t-SNE visualization on the features learned by the model, for each method AND each subset. (5 points)
    3. For each figure or table, you need to describe the observations and briefly explain them. (5 points)

2 Part 2: Wallpaper Classification with CNN

2.1 Introduction

In Part 2 of this project, we will build on the wallpaper images classification task and introduce a new approach using a Convolutional Neural Network (CNN) model for classification. You will also learn how to perform data augmentation for a more challenging test set.

2.2 Dataset: Wallpaper Pattern Images

You will be working with the wallpaper dataset. This dataset consists of images containing the 17 Wallpaper Group.
We have provided you with three wallpaper subsets: "train", "test", and "test_challenge".

- "train": it consists of (N) $17 \times 1,000 = 17,000$ wallpaper pattern images.
- "test": it consists of (N) $17 \times 200 = 3,400$ wallpaper pattern images.
- "test_challenge": it consists of (N) $17 \times 200 = 3,400$ wallpaper pattern images with more variations (random rotation, shifting, and cropping) to make it a challenging test set.

Each image is 256×256 and has 1 channel (grayscale). In other words, it has (D) $256 \times 256 = 65,536$ features. The dataset is composed of (C) 17 classes, each representing a unique wallpaper pattern group. See Fig. 1.

You are asked to train neural networks to classify these wallpaper patterns. The datasets are located in the "$data/wallpapers/ < train, test, test\_challenge > / < group > /$" folders. The train and test images are in separate folders and within them, there are folders for each wallpaper group's images.

Fig. 1 shows the images of 17 wallpaper pattern groups, that are used to form the "train" and "test" subsets. Fig. 2 shows the images with more variations, that are used to form the "test_challenge" subset.

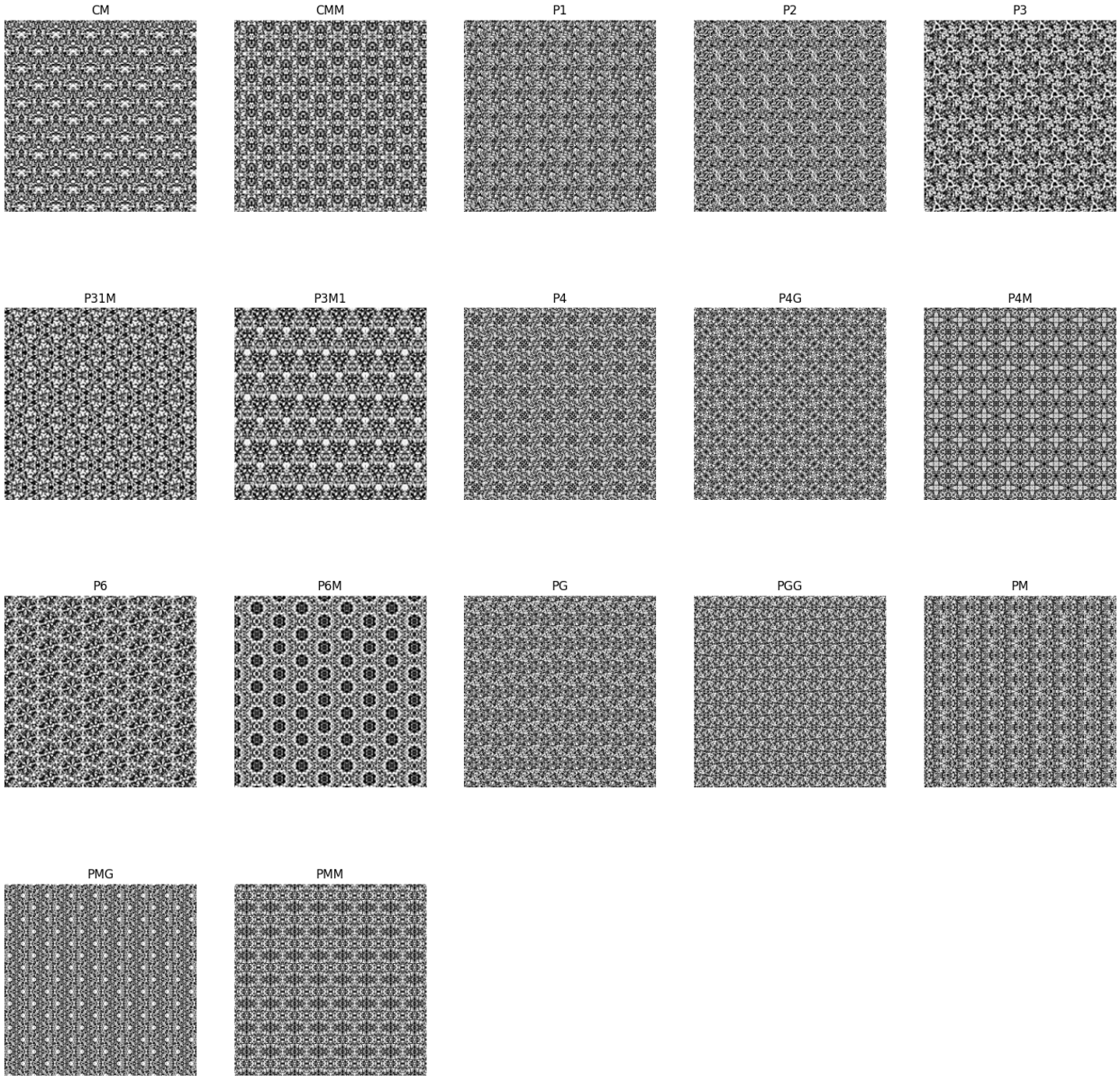

Figure 1: 17 Wallpaper Pattern Groups

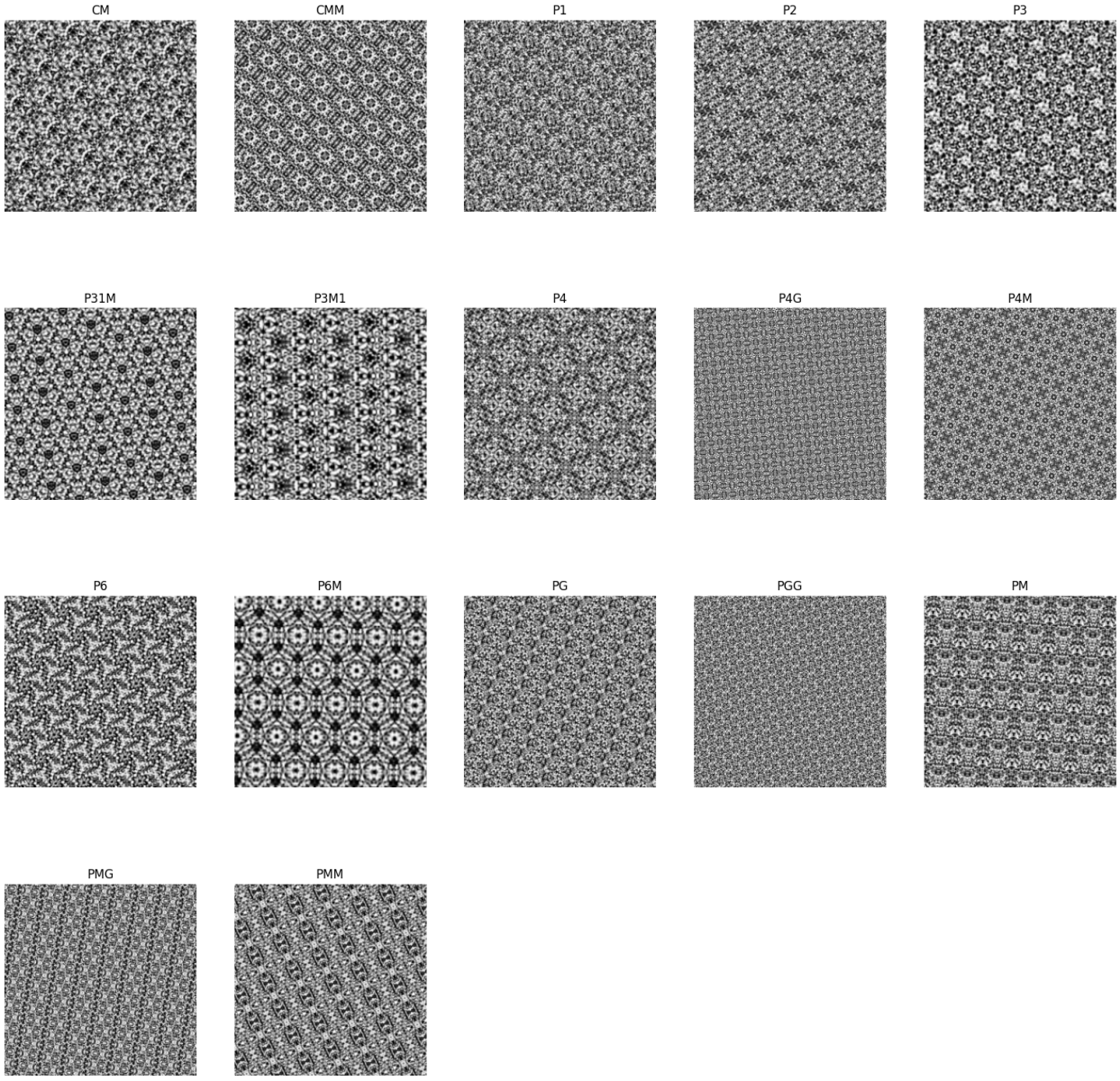

Figure 2: 17 Wallpaper Pattern Groups, more challenging with variations (random rotation, shifting, and cropping)

2.3 Implementation

To complete Part 2 of the project, you will need to train and evaluate a CNN model on the wallpaper dataset. We have provided a basic CNN architecture (Fig. 3) in the starter code, which you can use as a starting point. However, you will need to modify the layer settings to check the model's performance. You shall try at least two different layer settings and report the results of each.

Before training the model, it is important to adjust the training settings to suit your computer's performance.

Layer	Input Dimension	Output Dimension	Kernel Size	Stride	Padding	Activation Function
Conv2D	(batch, input_channels, img_size, img_size)	(batch, 32, img_size, img_size)	(3, 3)	1	1	ReLU
MaxPool2D	(batch, 32, img_size, img_size)	(batch, 32, img_size/2, img_size/2)	(2, 2)	2	0	None
Conv2D	(batch, 32, img_size/2, img_size/2)	(batch, 64, img_size/2, img_size/2)	(3, 3)	1	1	ReLU
MaxPool2D	(batch, 64, img_size/2, img_size/2)	(batch, 64, img_size/4, img_size/4)	(2, 2)	2	0	None
Conv2D	(batch, 64, img_size/4, img_size/4)	(batch, 128, img_size/4, img_size/4)	(3, 3)	1	1	ReLU
MaxPool2D	(batch, 128, img_size/4, img_size/4)	(batch, 128, img_size/8, img_size/8)	(2, 2)	2	0	None
Flatten	(batch, 128, img_size/8, img_size/8)	(batch, 128 x img_size/8 x img_size/8)	-	-	-	None
Linear	(batch, 128 x img_size/8 x img_size/8)	(batch, 1024)	-	-	-	None
Dropout	(batch, 1024)	(batch, 1024)	-	-	-	None
Linear	(batch, 1024)	(batch, num_classes)	-	-	-	None

Figure 3: The Baseline CNN Model Diagram

As part 1, you are asked to modify the architecture settings and aim at getting better performance than the baseline model we provided.

Additionally, you have to perform data augmentation techniques to improve the performance of your CNN model on the "test_challenge" subset. Data augmentation involves creating new training examples by applying various transformations to the existing images. In this part of the project, you should try a combination of data augmentation strategies to improve the performance of your model, including:

- **Rotation** in 360 degrees.
- **Uniform scaling** between 100% -200% from the original image.
- **Valid translation** in any direction without going out of the image.
- Flipping

- Cropping

You can see more details in https://pytorch.org/vision/stable/transforms.html. You shall report how the data augmentation techniques are applied, and how data augmentation affects the performance of your model.

2.4 Requirements

- Train the baseline model and evaluate its performance on the 'train", "test", and "test_challenge" subsets, respectively.
- Implement data augmentation and improve the baseline model to achieve better performance than the baseline model. Evaluate the performance of your new method on the 'train", "test", and "test_challenge" subsets, respectively.
- To evaluate the performance of each method AND each subset, you must:
  - Display classification results (confusion matrix, overall accuracy & standard deviation, and the classification accuracy for each class) in either figures or tables.
  - Visualize the feature maps of one convolutional layer. You must state which layer you choose for visualization. Make sure to choose the same layer for all evaluations. Provide figures to support your findings and describe your observations.
  - Perform t-SNE visualization on the features learned by the model. You must perform t-SNE visualization on the last fully connected layer of your model for all evaluations. Provide figures to support your findings and describe your observations.

2.5 Grading Criteria of Part 2 (70 Points)

- Train and Test the Baseline Model (**5 points**)
  1. Achieve reasonable performance on the "train" and "test" subsets (no requirement for "test_challenge"). (5 points)
- Train and Test You Own Model (**30 points**)
  1. Perform data augmentation for training. (10 points)
  2. Achieve better performance on the 'train", "test", and "test_challenge" subsets than the baseline model. (10 points)
  3. Code implementation. (10 points)
- Report Part (**35 points**)
  1. Display classification results for each method AND each subset. (5 points)

2. Visualize the feature maps of one convolutional layer, for each method AND each subset. (10 points)
3. Perform t-SNE visualization on the features learned by the model, for each method AND each subset. (10 points)
4. For each figure or table, you need to describe the observations and briefly explain them. (10 points)

3 Extra Credits (Up to 50 Points)

In this project, we encourage students to explore and experiment with different model designs and training techniques to achieve better performance on the Taiji key-pose classification and Wallpaper pattern classification tasks. To encourage students to go above and beyond, we will provide extra credit for exceptional work. Specifically, we will give bonuses to students who satisfy any of the following:

- Superior performance: achieve higher classification rates, particularly for the challenging "test_challenge" subset of the wallpaper pattern classification task.
- Creativity: come up with a unique model architecture, reduce training time, improve inference speed, or produce a smaller model size while maintaining reliable performance, among other possibilities.
- Thorough exploration: conduct a comprehensive analysis of various model architectures or training hyper-parameters to provide a detailed report on how they affect performance.
- Novel approach: propose and implement a new approach that significantly improves the performance or the efficiency of the existing models.

We encourage all students to strive for excellence and take advantage of this opportunity to showcase their creativity, innovation, and mastery of the course concepts.

4 Submission

Your submission should include the following items:

- Your code, which should be **reasonably commented** on and written in an understandable manner.
- Your written report, which should be following the requirements & criteria listed above.
- A ReadMe document that explains the function of each code file and which data file is used.

- Please do not include the data file in your submission.

Note that including the data file in your submission is unnecessary and may cause issues with file size and upload time.
Please package all of these items into a single zip file and name it as

FirstName_LastName_ProjectNo.zip.

For example, **John_Doe_3.zip**.

Make sure to properly cite all resources you used for this project and double-check your submission before uploading it to the Canvas dropbox. Please note that graders will read and run your code, so please ensure that your code runs correctly and is well-organized and easy to understand.
Be sure to fully read the README.md.

5 Common Issues

Notice that this section will be updated if more common issues are asked about by the students.