# Project 3: Deep Learning with an introduction to MLP and CNN

TAs: Shimian Zhang, Keaton Kraiger
02/21/2023

# Summary of Project 1

# Good Organization

Contents

# Nice Figures

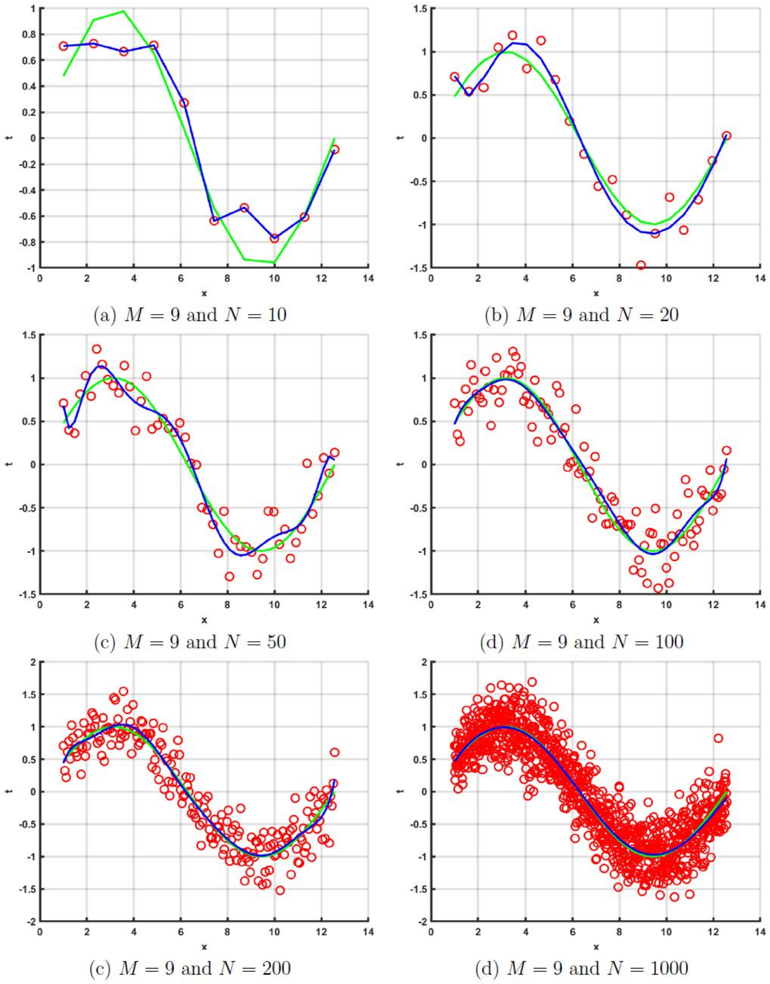


Figure 3: Plots of the solutions obtained by minimizing the sum-of-squares error function using different $N$ values

## Nice Tables

| | $m=1$ | $m=2$ | $m=6$ | $m=9$ | $m=10$ |
|---|---|---|---|---|---|
| $w_0$ | 1.1658 | 1.6174 | -1.0869 | -14.8321 | -6.5722 |
| $w_1$ | -0.1738 | -0.3519 | 2.1414 | 35.2255 | 12.1939 |
| $w_2$ | | 0.0131 | -0.8360 | -32.0299 | -6.1481 |
| $w_3$ | | | 0.1672 | 15.5251 | -0.0805 |
| $w_4$ | | | -0.0202 | -4.4425 | 1.2130 |
| $w_5$ | | | 0.0013 | 0.7848 | -0.5164 |
| $w_6$ | | | -3.33e-5 | -0.0865 | 0.1079 |
| $w_7$ | | | | 5.7882e-3 | -0.0129 |
| $w_8$ | | | | -2.1500e-4 | 9.1615e-4 |
| $w_9$ | | | | 3.399e-6 | -3.5178e-5 |
| $w_{10}$ | | | | | 5.6874e-7 |

Table 1: Table of the fit coefficients $\mathbf{w}_{\mathrm{ML}}$ (ML) for polynomials of various order.

# Introduction of Project 3

- This project is about applying Deep Learning techniques to **Taiji key-pose classification** & **Wallpaper pattern group classification** tasks.

- Learn how to train and test deep learning models including two basic architectures: MLP and CNN

- Release Date: Tuesday, Feb 21

- Submission Due Date: **Monday, March 13**

*Python starter code is provided.*

# Part 1: Taiji Keypose Classification with MLP

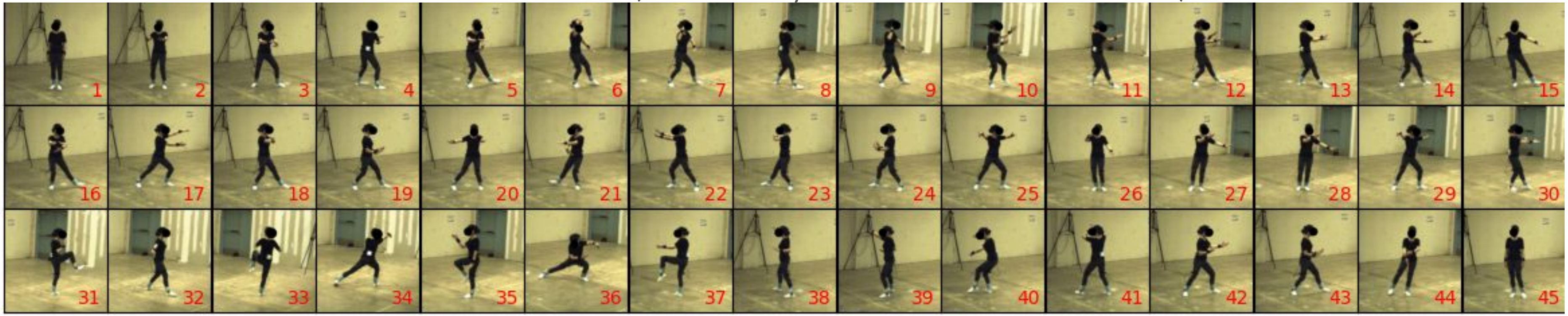
- Dataset: PSU-TMM100
  - N: 49,774 samples.
  - C: 45 + 1 classes.
  - D: **1,961 features** – *full version* **67 features** – *lod4 version*

- Deep Learning Model: Multi-Layer Perceptron (MLP)

# PSU-TMM100: Classes

Each frame is associated with one of the following classes:

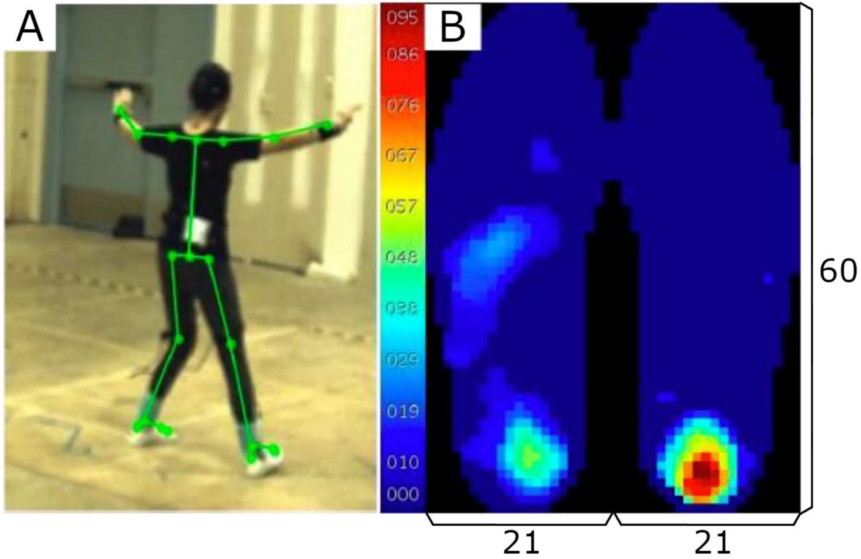
- 0 -- NON KEY FRAME (frames in-between two KEY forms).
- 1-45 -- ONE OF THE KEY FORMS (names of key forms included in the dataset)



# PSU-TMM100: Features (full Version)

The Taiji data consists of two kinds of features:

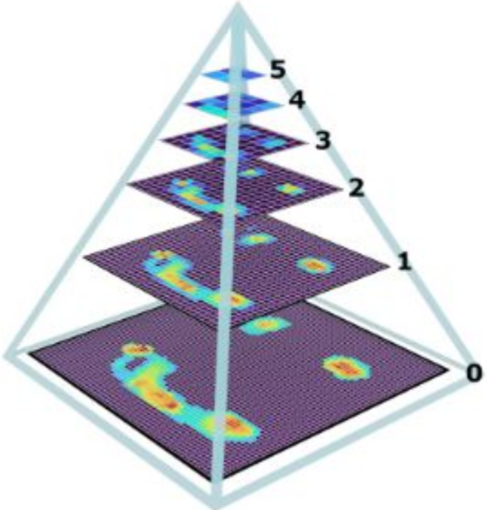
- 3D body joints feature from MoCAP (Motion Capture)
  - 17 (Joints) x 3 (dimensions) = **51** features
- Foot Pressure data
  - 60 (height) x 21 (width) x 2 (feet) = 2520
  - After cleaned-up (mask processed), the 2520 features are reduced to **1910** features.
- Total # of data dimension : D = 1,961



# PSU-TMM100: Features (lod4 Version)

The Taiji data consists of two kinds of features:

- 3D body joints feature from MoCAP (Motion Capture)
  - 17 (Joints) x 3 (dimensions) = **51** features
- Foot Pressure data after 4-levels of downsampling
  - 4x4= 16 features.
- Total # of data dimension : D = 67




Foot pressure downsampling

# Implementation

- You will need to train and evaluate an **Multi-Layer Perceptron (MLP)** model on the BOTH Taiji datasets. (*Baseline model is provided*)

- After practicing the baseline model, you have to modify the network's architecture (layers, activations, etc.) and seeing the model's performance.

- You shall try at least one modified architecture and report results.

# Multi-Layer Perceptron (MLP)

- MLP is a type of neural network that consists of multiple layers of interconnected nodes
- An MLP consists of one or more hidden layers, an input layer, and an output layer.
- The MLP learns to adjust nodes' weights & biases between the predicted output and the actual output. (**Back propagation** )

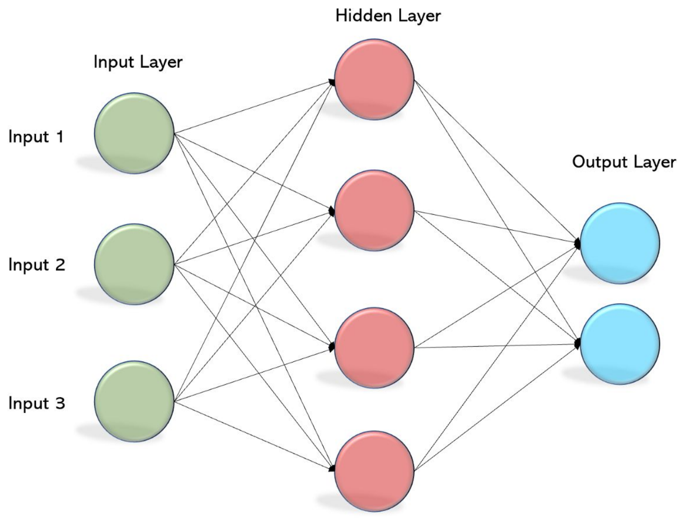



*Figure from [Multi layer Perceptron (MLP) Models on Real World Banking Data]*

# Multi-Layer Perceptron (MLP)

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

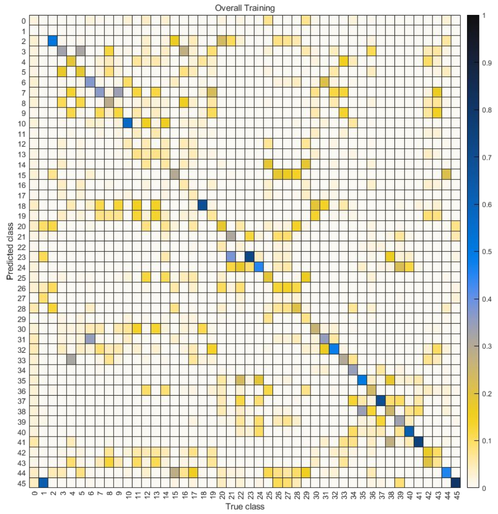
Table 2: Provided basic MLP architecture.

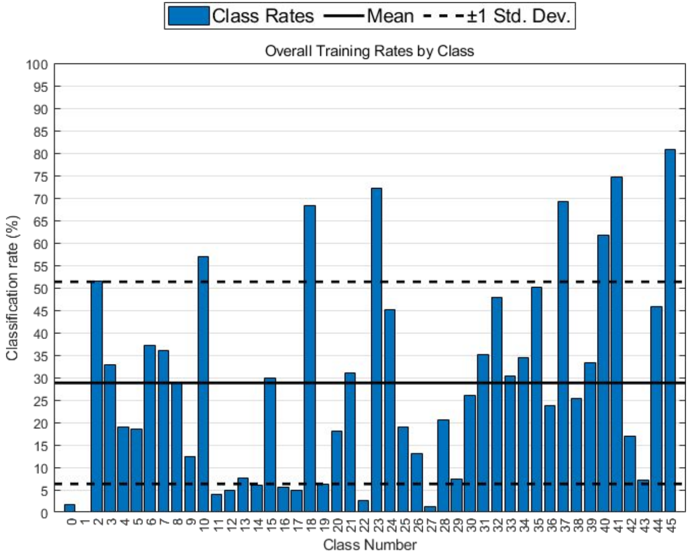
# Evaluation Requirements

- You will have 2 models to evaluate: (1) Baseline and, (2) Updated architecture
- There are 2 datasets to train & test: full version & lod4 version
- For each model-dataset combination, you need to report the following:
  - **Training & Testing** overall accuracy **& std**, averaged on all LOSO iterations.
  - **Test** classification accuracy & std for each class, averaged on all LOSO iterations.
  - Confusion matrix of a **test subject of your choice**. The same subject must be used for all evaluations.

# Evaluation Requirements

- Confusion matrix and classification rate



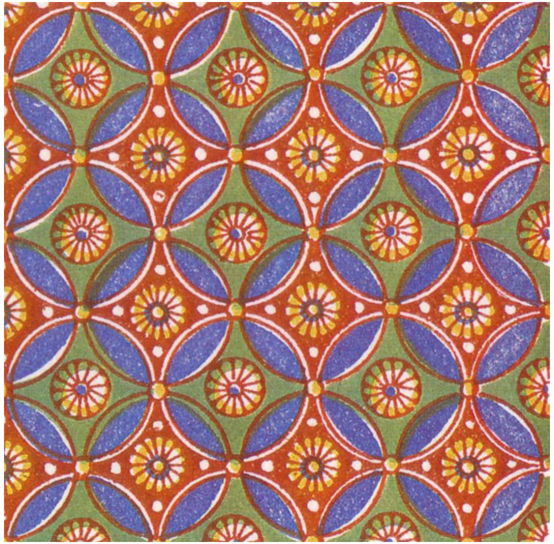



# Part 2: Wallpaper Classification with CNN

- Dataset: Wallpaper Pattern Images
  - This dataset consists of images containing the 17 Wallpaper Group.
  - You will be provided with 3 subsets:

    "train", "test", and "test challenge"

- Deep Learning Model: Convolutional Neural Network (CNN)



Example of an Egyptian design with wallpaper group **p4m**

*Figure from Wallpaper Group*

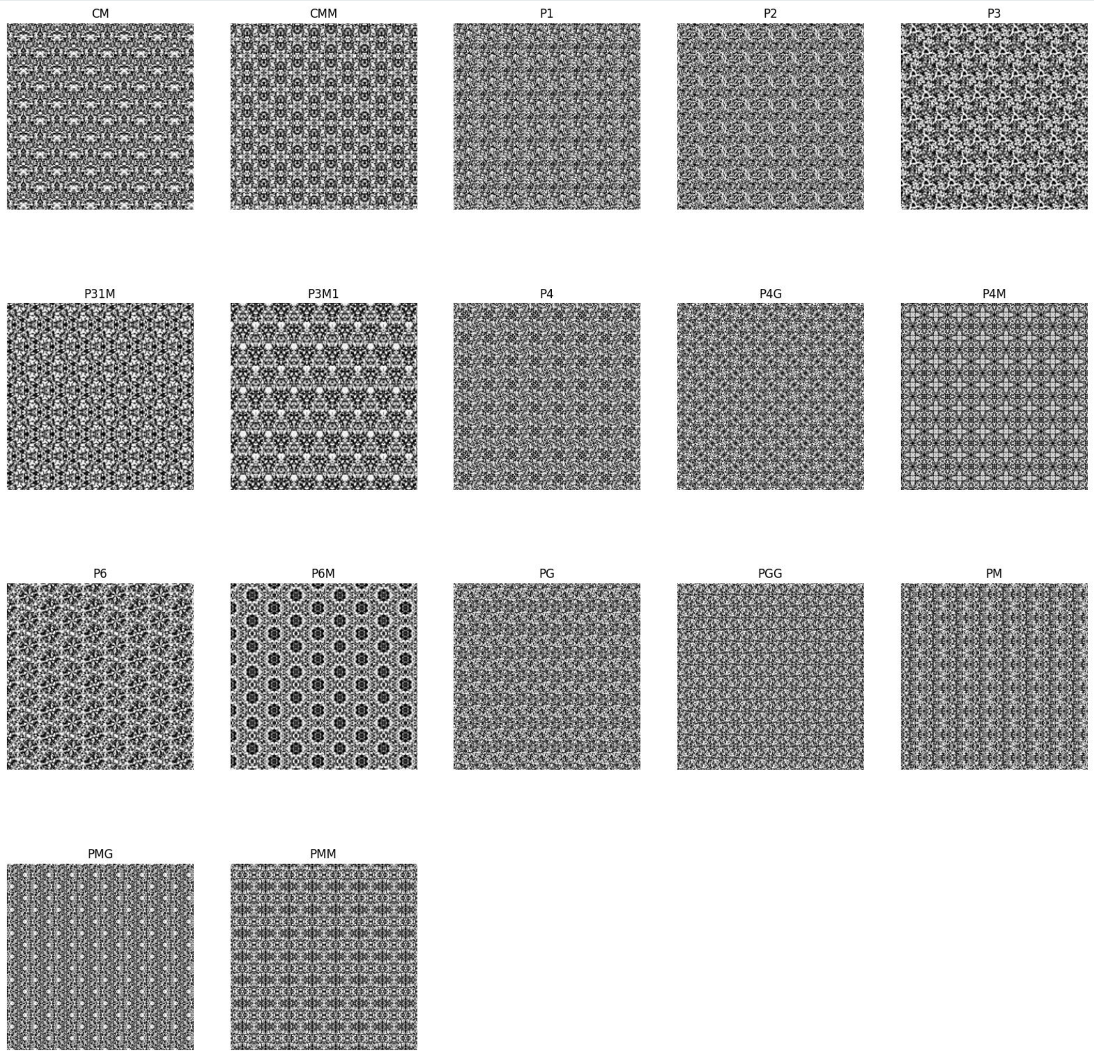
# Wallpaper Pattern Images

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.

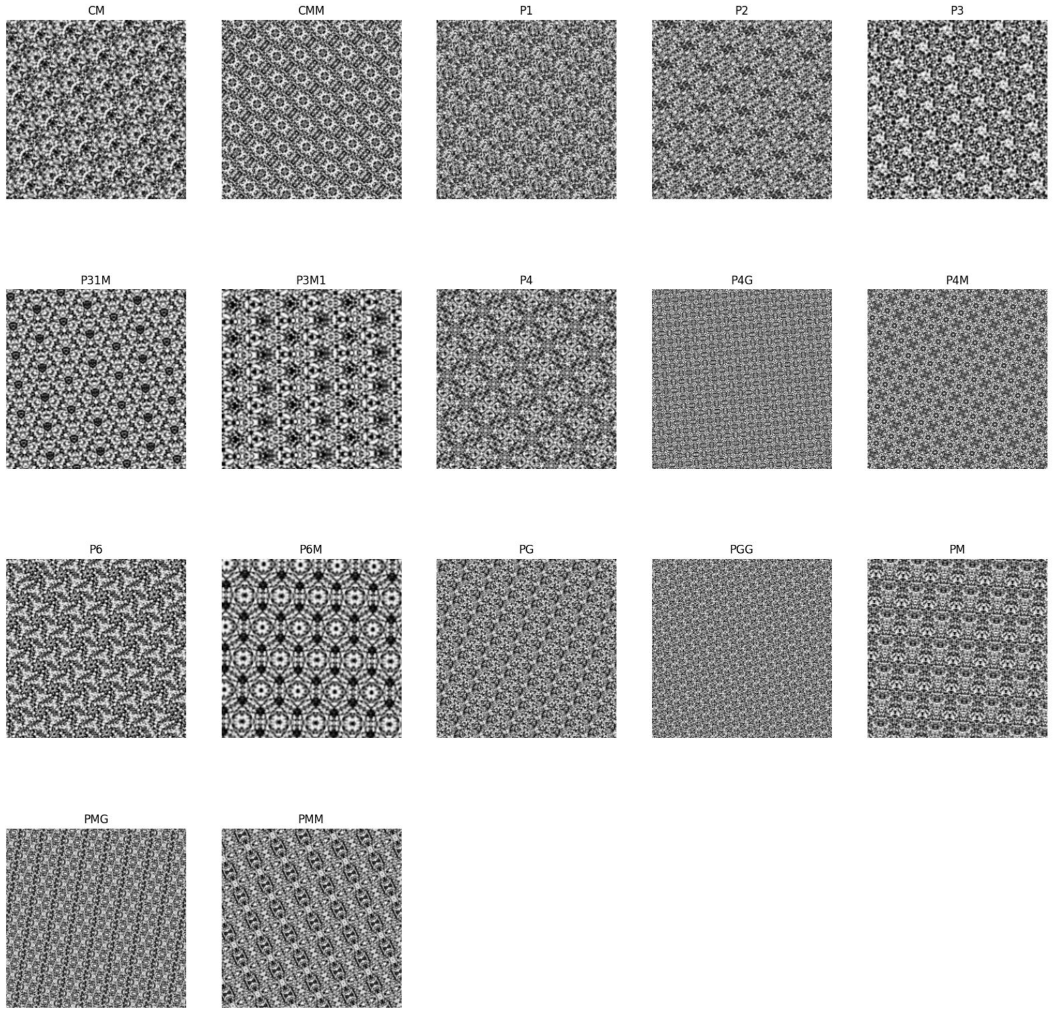
Each image is $256 \times 256$ and has 1 channel (grayscale). In other words, it has ($D$) $256 \times 256 = 65,536$ features. The dataset is composed of ($C$) 17 classes, each representing a unique wallpaper pattern group. See Fig. 1.

# Wallpaper Patterns



# With Variations

*Picked from the “test_challenge” subset*



# Convolutional Neural Network (CNN)

- Convolutional neural networks (CNNs) are a type of deep learning model commonly used for image classification, object detection, and other computer vision tasks.
- In a CNN, the input image is processed by a series of **convolutional layers**, each of which applies a set of filters to extract different features from the image. The outputs of these layers are often followed by **pooling layers**, which downsample the feature maps to reduce their dimensionality.

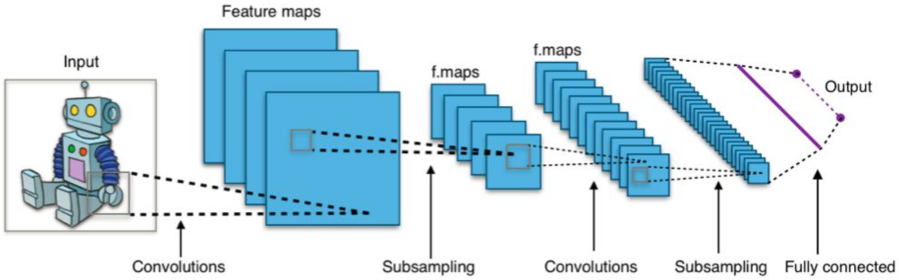



*Figure from Convolutional neural network*

# Convolutional Neural Network (CNN)

| 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

# Data Augmentation

- To achieve reasonable performance on the “test_challenge” subset, you are required to perform data augmentation.
- Data augmentation is a technique used to increase the size of a training dataset by artificially creating new data.
- Common data augmentation techniques include:
  - Flipping and rotating images.
  - Cropping or resizing image.
  - Adding noise or distortion to images.
  - Changing brightness or contrast levels.

# Evaluation Requirements - 1

- You will have 2 models to evaluate: (1) Baseline and, (2) Updated architecture
- For each model, you will need to **first** train it with the provided "train" subset, and report the following:
    - Training and testing ("test", and "test challenge") overall accuracy **& std**
    - **Test** ("test" only) classification accuracy & std for each class.
    - **Test** ("test" only) confusion matrix for each class.

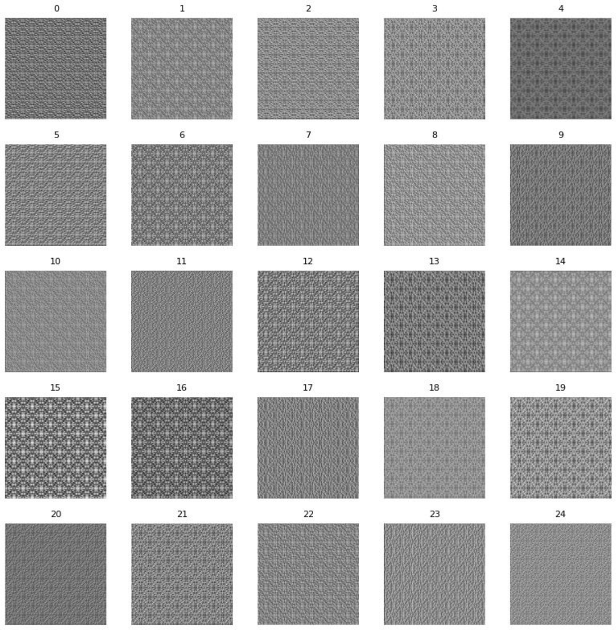
## Evaluation Requirements - 2

- Then, you need to perform data augmentation with the updated model, and report the following:
  - Training (after augmentation) and testing ("test", and "test challenge") overall accuracy **& std**
  - **Test** ("test challenge" only) classification accuracy & std for each class.
  - **Test** ("test challenge" only) confusion matrix for each class.

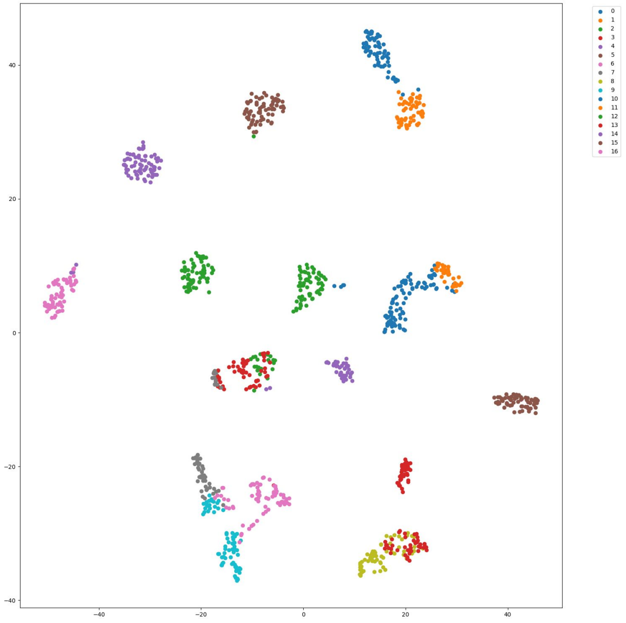
# Feature Visualization

To gain insight into how well your CNN model has learned features, you are required to perform the following two visualizations after training your own model:

- Feature visualization:
  - Visualize the feature maps of a chosen convolutional layer with 17 images from 17 wallpaper groups.
  - **Specify which layer you choose for visualization**.
- t-SNE visualization:
  - Perform t-SNE visualization on the last fully connected layer of the model for all evaluations.




Example of feature map




Example of t-SNE

# Extra Credit:
# A Class Competition on DL-based 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 will offer 50 points to the winner in each category mentioned above.**

# Some Tips

- Train and test on GPUs if available.
  - We will release a survey this week to see how much computation resources we may request from the department.
- Always save checkpoints during training & save the trained models.

# Q & A