# Introduction

A convolutional neural network is a type of neural network which is generally used in applications where the size of the dataset is very large. Due to this, it is computationally very intensive to have all hidden layers fully connected. As a result, in a convolutional neural network, each neuron is only connected to a locally contiguous subset of nodes from the previous layer. It consists of two types of layers, convolutional layer and pooling layer. Each convolutional layer has multiple feature maps, where each feature map is a 2D arrangement of neurons used to calculate a single feature, whose value is a 2D matrix. Each convolutional layer is connected to a pooling layer, which is used to reduce the size of the feature maps in the previous layer by dividing it into disjoint regions and pooling each region to a single scalar value. This kind of network is widely used in computer vision for image and video recognition.

# Strategies

In order to have a fair comparison, we make sure that we use the same training and test set for all the models. We plot the training accuracy against the number of epochs for each model and also the test accuracy against different values of each parameter. We choose the parameter values which give the highest test accuracy. Note that we do not use the training accuracy to choose the parameter values, since the training accuracy will be high although the test accuracy will be low if the model overfits the training data. Due to this, such a model will not generalise well to new examples.

# Methodologies

## Model Architectures

### 1 Convolutional and 1 Pooling Layer

In this architecture, we experiment with the best values for the learning rate, mini-batch size, momentum and decay term.

### 2 Convolutional and 2 Pooling Layers

In this architecture, we use the learning parameters obtained from Part 1 and experiment with the best values for the number of filters for both the layers.

## Learning Algorithms

We use stochastic gradient descent with momentum for training the models. In this algorithm, the training set is divided into multiple batches of equal size and the weights are updated for one batch at a time. Backpropagation is used to implement gradient descent. Momentum is used for speeding up the convergence and ensuring that the algorithm does not oscillate for a long time near the optimum. Decay term is used to ensure that the model does not overfit the training data.

# Implementation

The file “cnnPreprocess.m” is used to randomly divide our dataset into training set and test set. The same division of data is used to experiment for all the models and architectures.

The file “cnnTrain1” is used to train the architecture for Part 1 for different learning parameters. The file “cnnTrain2.m” is used to train the architecture for Part 2 for different number of filters. We also have separate files for varying the parameters to make it more modular –

1. vary\_lr.m: This file is used to experiment with different learning rates.
2. vary\_batchsize.m: This file is used to experiment with different batch sizes.
3. vary\_momentum\_decay.m: This file is used to experiment with different combinations of momentum and decay terms.
4. vary\_num\_filters.m: This file is used to experiment with the number of filters for both the convolutional layers in Part 2

Once we have found the optimum parameters for both the architectures, we find their accuracy on the test set to evaluate how well the algorithm performs for those parameters.

# Results and Analysis

## Model with 1 Convolutional and Pooling layer

### Learning Rate

#### Reconstruction

// Compare best and worst MSE

#### MSE

### Batch Sizes

#### Reconstruction

#### MSE

### Momentum and Decay

#### Reconstruction

#### MSE

## Model with 2 Convolutional and Pooling layers

### Number of filters

# Discussions and Challenges