Image Classification Using CNN from Optimization Perspective

Farida Far Poor, Mohammad Mehdi Hosseini

Table of contents

Classification

Classification:

- Design a model
- Look at many samples (providing data)
- Learn features (loss minimization)
- Label new **unseen** data

Classifier is

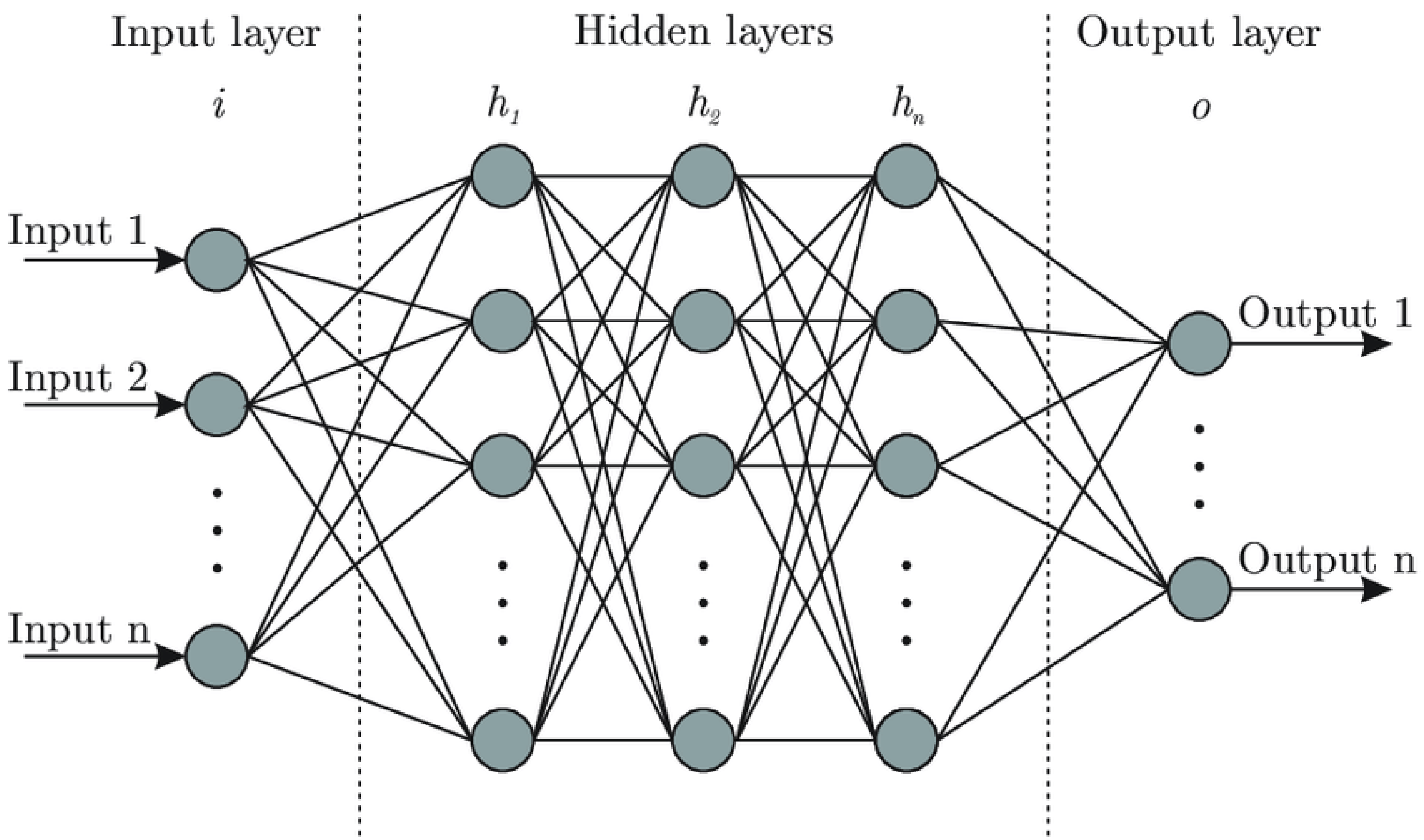
- a **model** that produces **labels** from a set of **features** of a **data**

Image classifier:

- Model
- Utilize Data (images)
- Train itself
- Label data

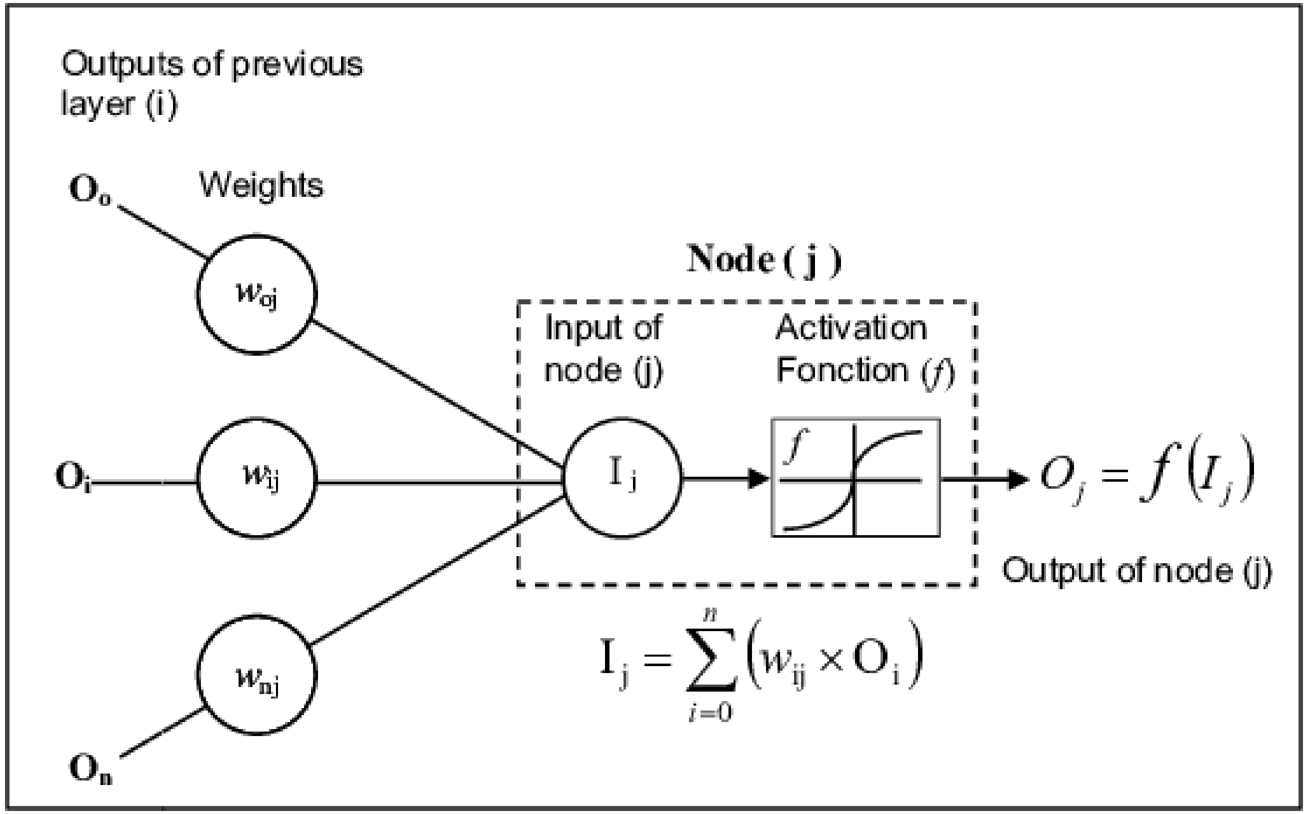
Classic Neural Network

Feed forward + Back-propagation



Classic Neural Network

Inside a node



Traditional Methods Drawbacks

Main problem in classification:

Feature extraction is

boring

time consuming

not exact

Features like: HOG, SIFT, HSV, RGB are understandable by human, not necessarily by the computers.

Classic Neural Network

Other assistants:

Feature selection

Feature generation

Painkillers

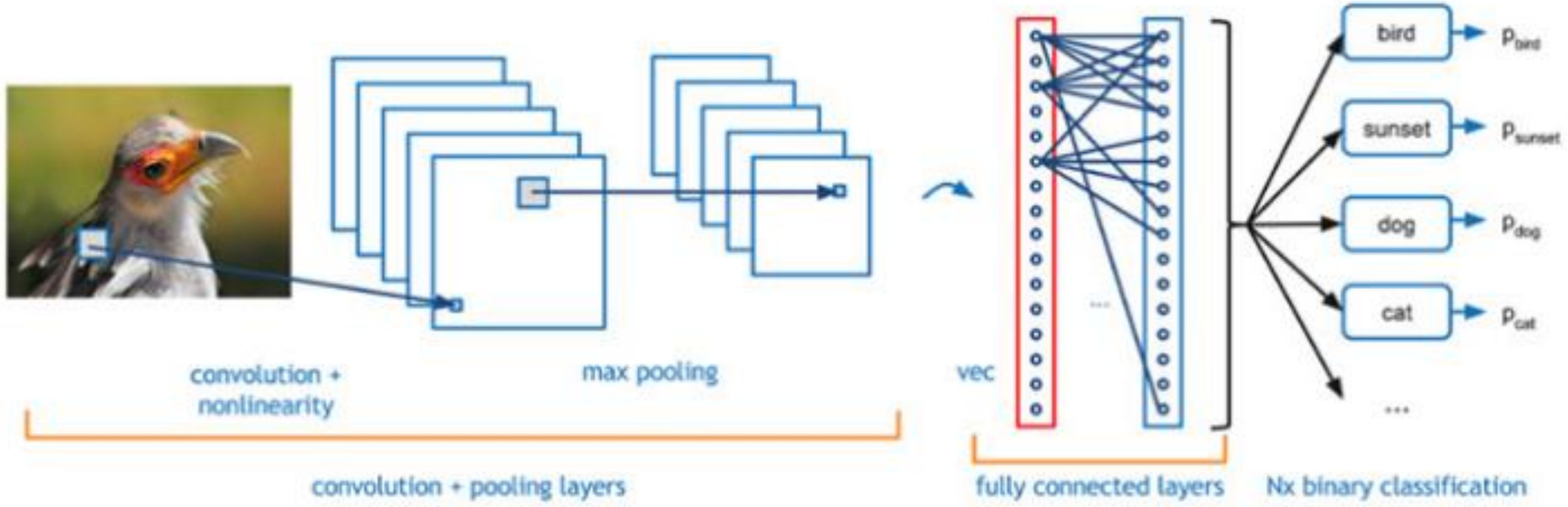
Question

Computers' understanding is not quite the same as humans,

Isn't it better let them do that?

Now, we know why we use convolutional neural networks

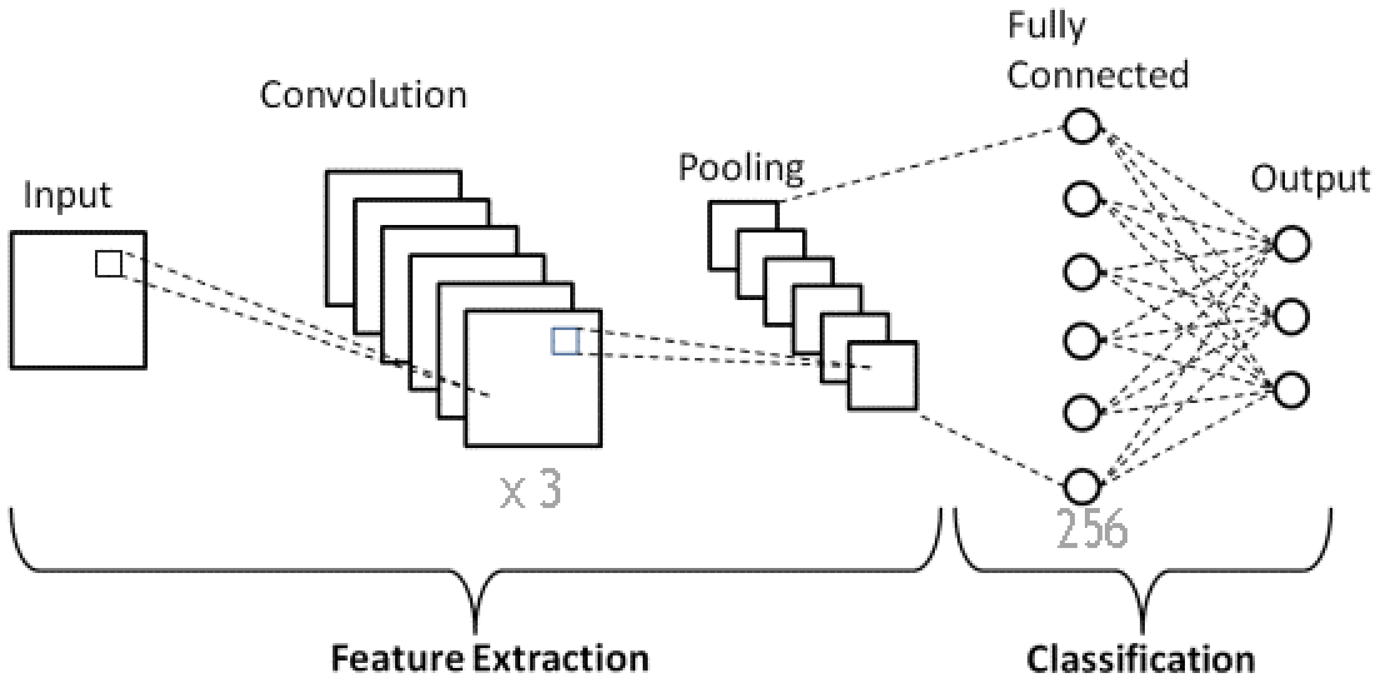
Convolutional Neural Network



Network Structure

3 convolutional layers

1 fully connected layer

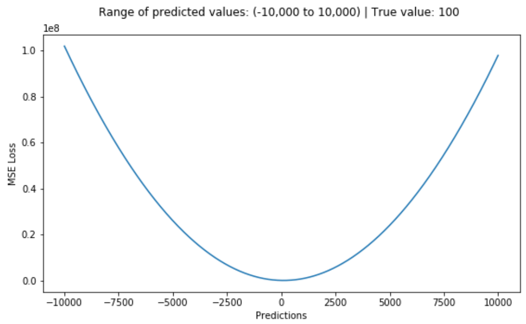


Optimization Perspective

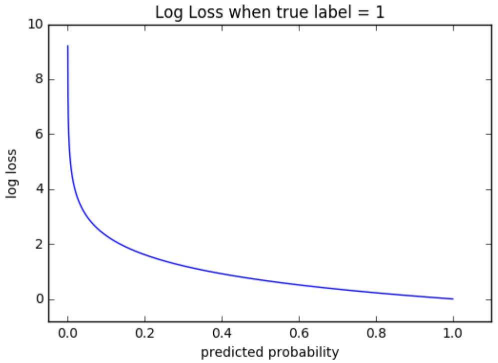
- Maximize Accuracy (Minimize Error)
  - Loss Function
  - Optimizer
    - Learning Rate
  - Activation function

Loss Function

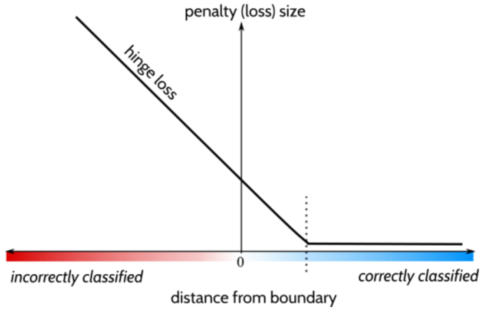
✓MSE



✓Cross-entropy



✓Hinge



Mean Squared Error (MSE)

General Form:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$$

MSE (suitable for regression):

$$f(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i(\boldsymbol{\theta}))^2$$

For example, in a 2D space:

$$\min_{w_0,w_1} \frac{1}{n}\sum_{i=1}^{n}(y_i - (w_0 + w_1 \cdot x_i))^2$$

w1 → slope   w0 → intercept

Cross-Entropy

General Form:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta)$$

Cross entropy (binary classifier):

$$\text{Loss} = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)\right)$$

Categorical Cross Entropy (Multi-class classifier):

$$\text{Loss} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C} y_{i,j} \cdot \log(\hat{y}_{i,j})$$

CE is a 0/1 classifier.

Hinge

General Form:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

Hinge (binary):

$$\text{Hinge Loss} = \frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i \cdot f(\boldsymbol{x}_i)\right)$$

Hinge with constraint:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left(\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i \cdot f(\boldsymbol{x}_i; \boldsymbol{\theta})\right)\right) + \lambda\|\boldsymbol{\theta}\|^2$$

Hinge loss is a -1/1 classifier.

Optimizers

Stochastic Gradient Descent (SGD)

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t; x^{(i)}, y^{(i)})$$

θ_t → Parameters at time t

η → Learning rate

$\nabla J(\theta_t; x^{(i)}, y^{(i)})$ → Gradient

Works better using batches.

Optimizers

ADAM

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$$

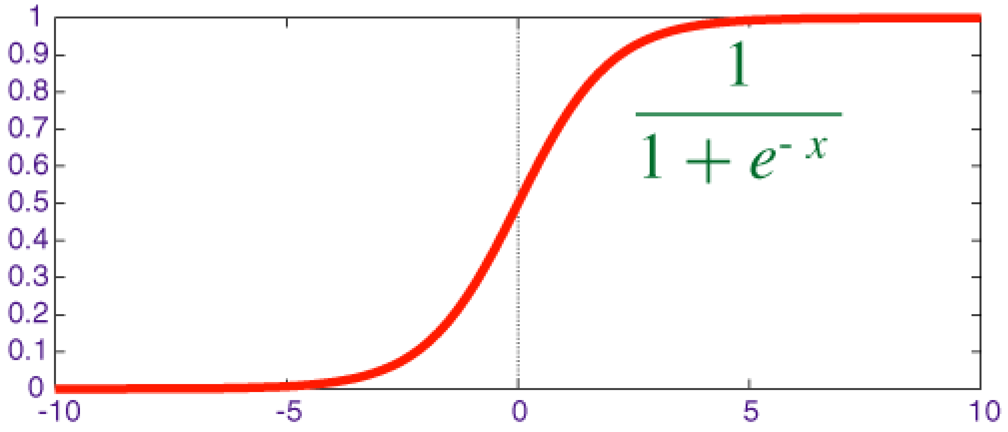
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla J(\theta_t)$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla J(\theta_t))^2$$

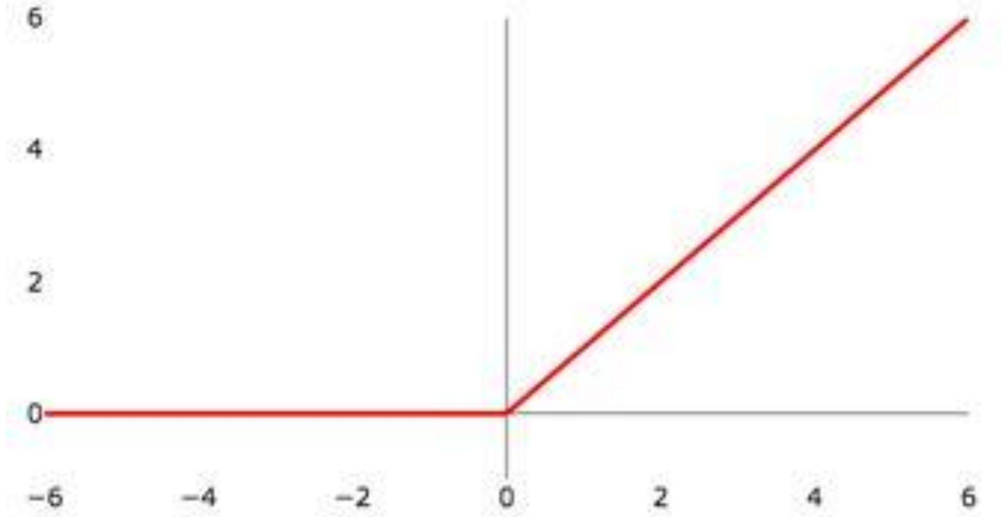
How to choose learning rate?

Activation Function

Sigmoid



ReLU



Experiments

- Classification on two datasets:
  - Caltech 101
  - Cifar 100
- Study network performance:
  - Different loss functions
  - Different optimizers
  - Different activation functions
  - The best learning rate

Baseline

Dataset: Caltech

Train-test ratio: 70-30%

Loss: categorical crossentropy

Optimizer: adam

Learning rate: 0.00003

Activation function: ReLU

Pooling: max

Batch size = 32

Epoch: 20

Datasets

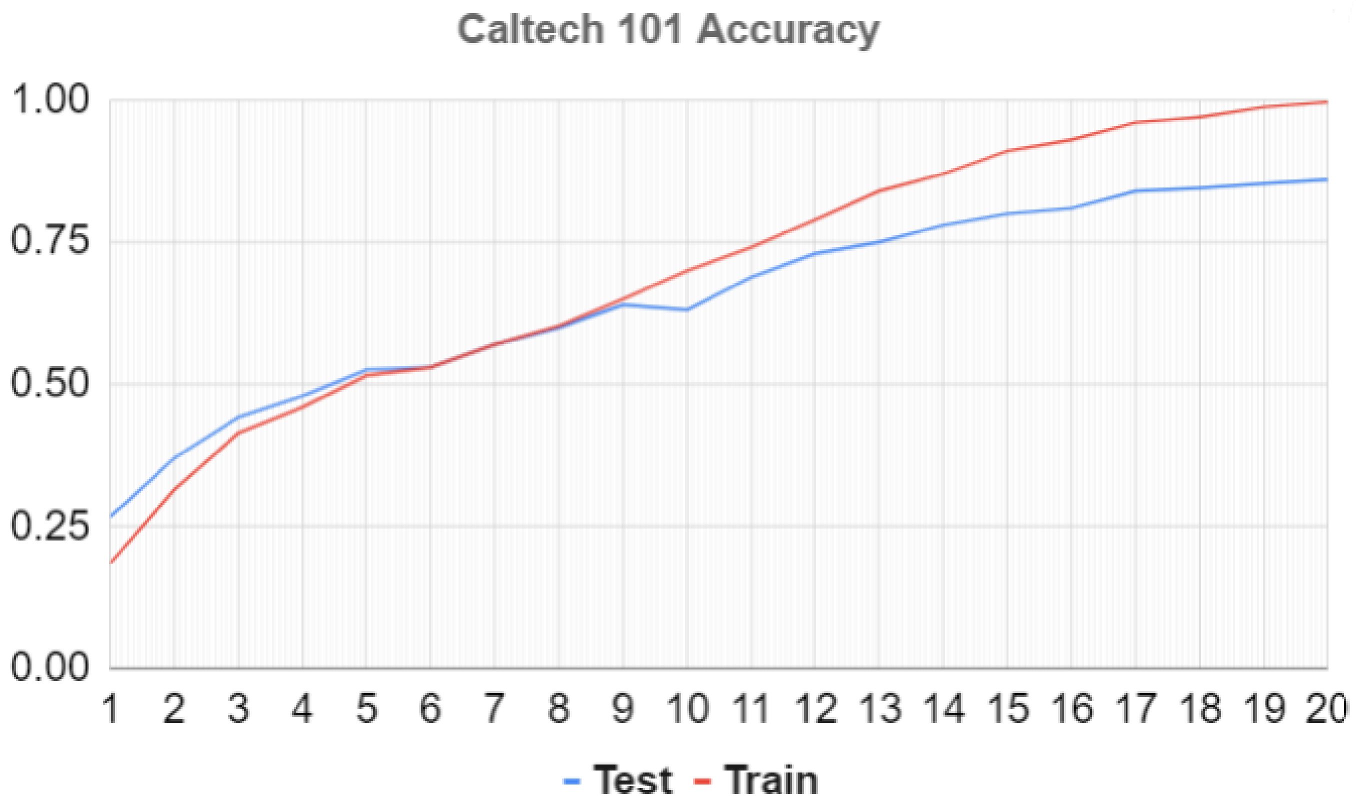
1. **Caltech 101**
   102 classes
   9000 images
   each class 40-800 images
   different image size (200~300)

2. **Cifar 100**
   100 classes
   60000 images
   each class 600 images
   32x32

Caltech
Accuracy on Model

Image Size: 256x256

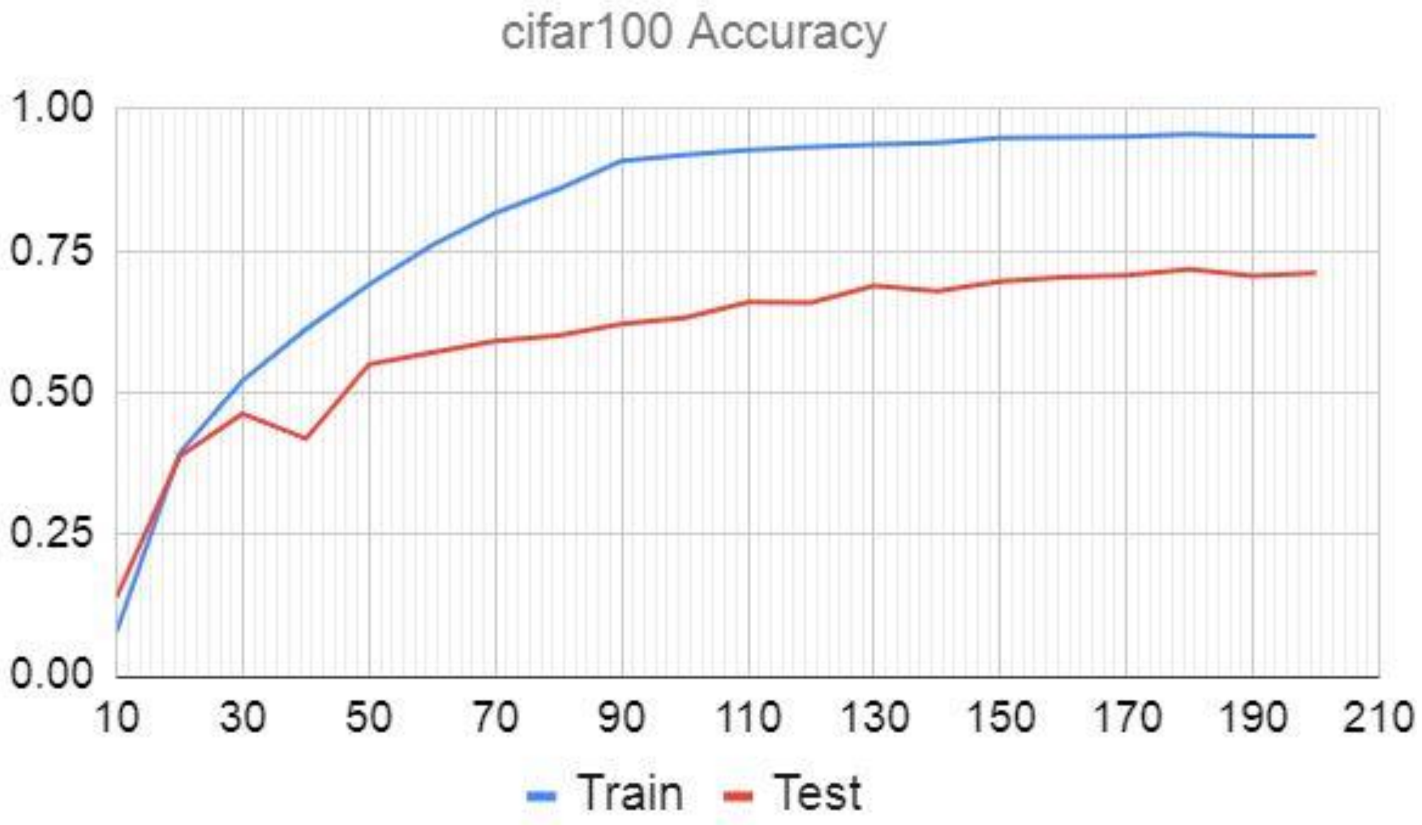
Parameters ~ 15 M



CIFAR
Accuracy on Model

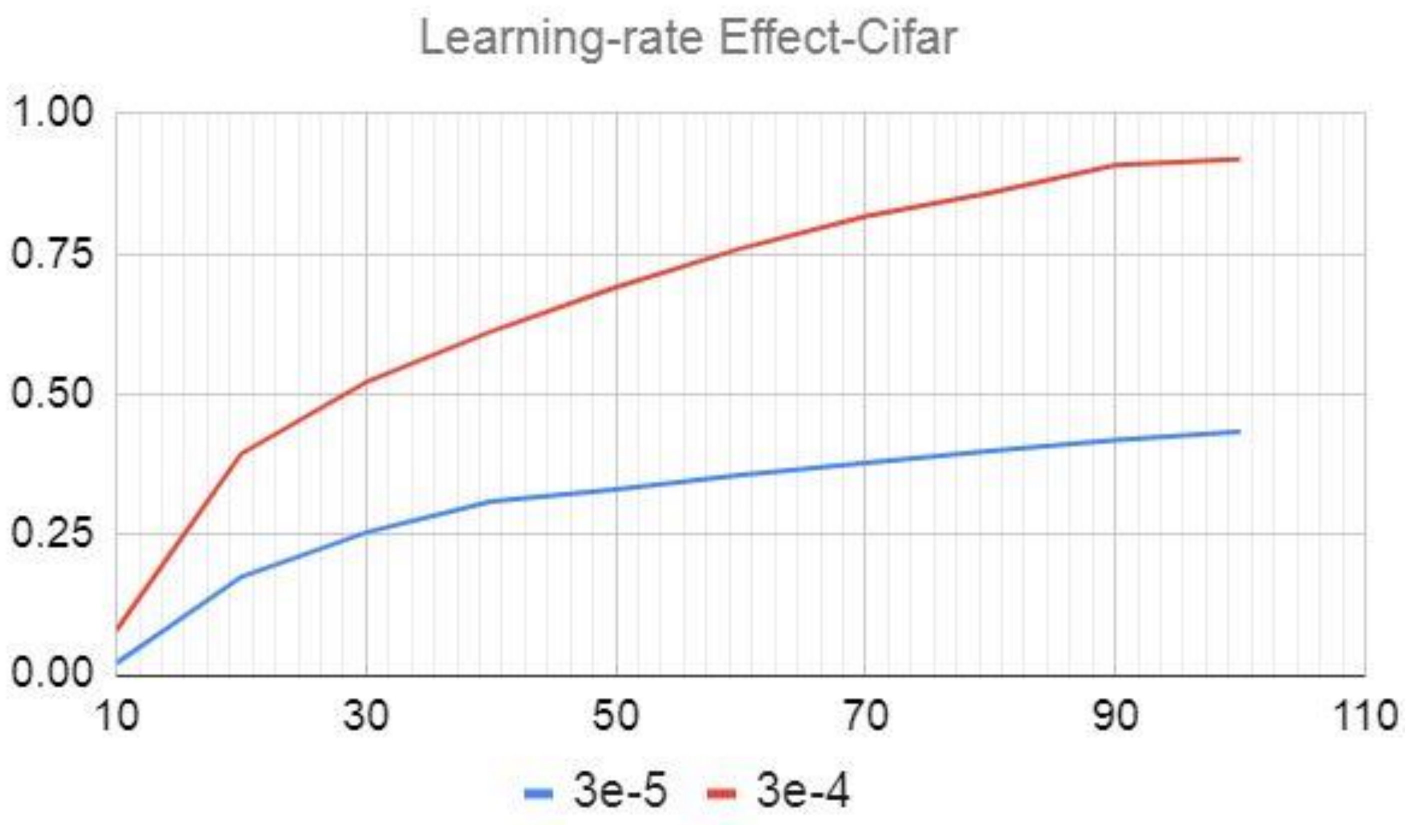
Image Size: 32x32

Parameters ~ 250 K



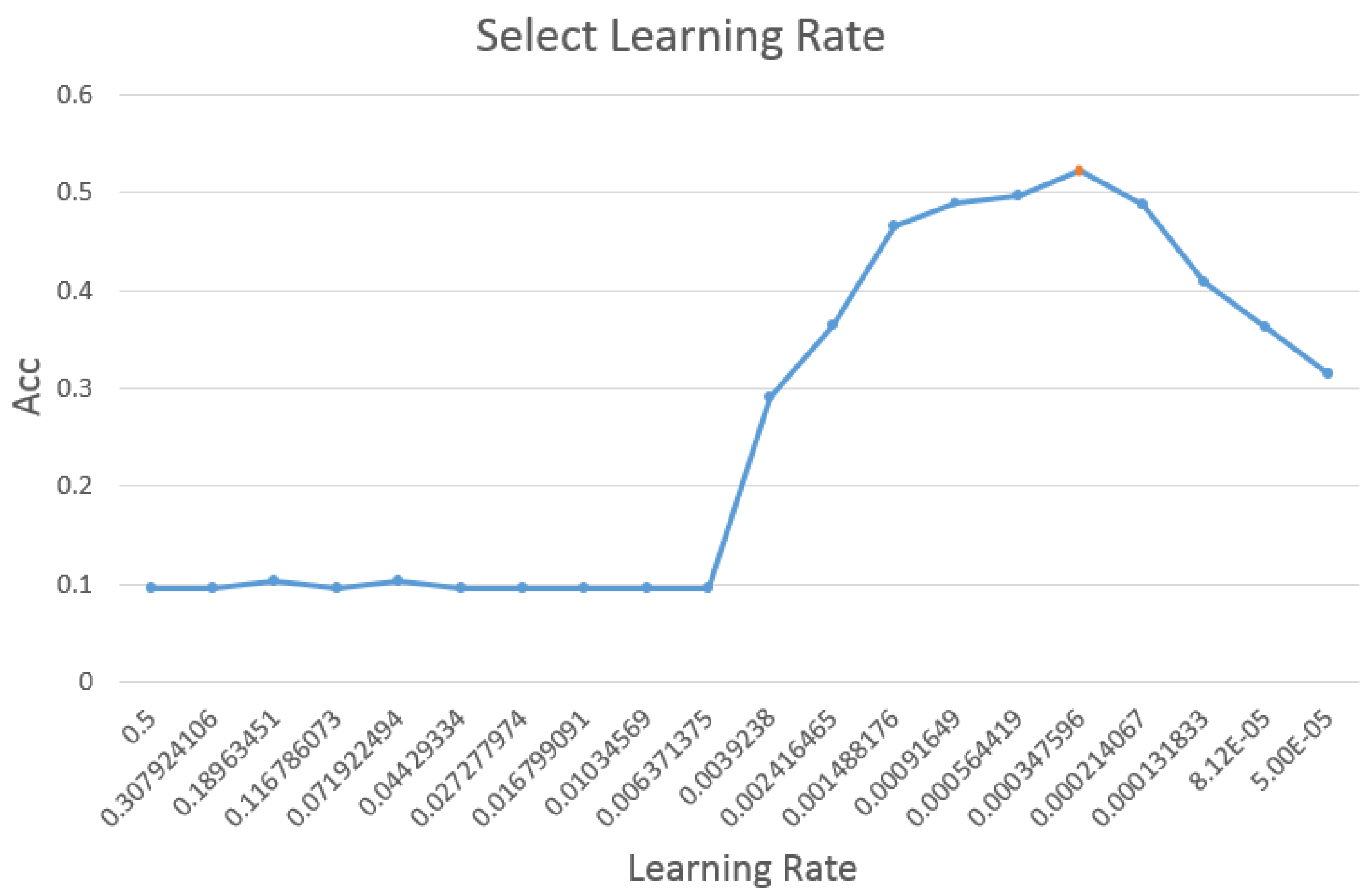
Learning Rate

Adam optimizer



Learning Rate (Caltech)

Adam optimizer



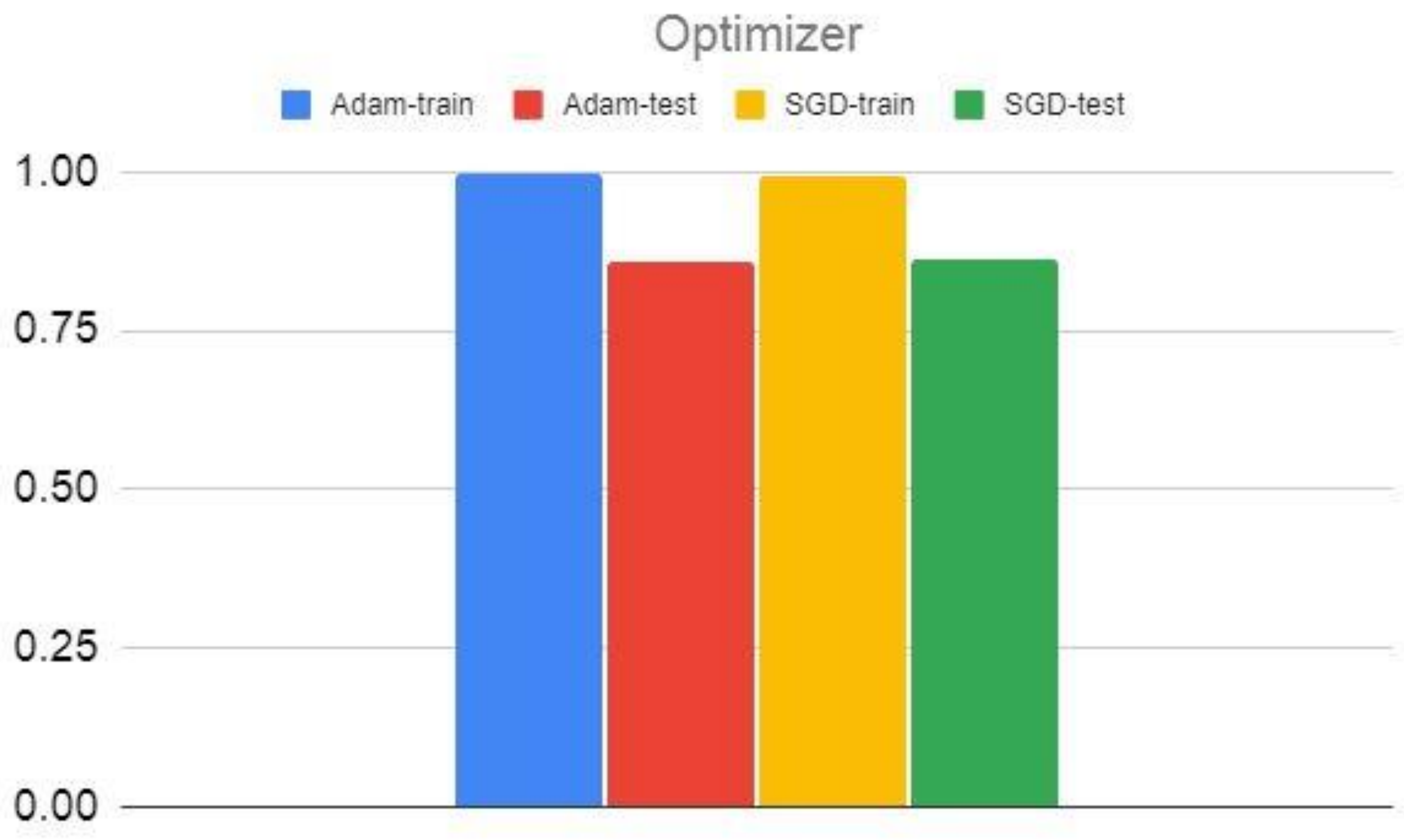
Learning Rate (Cifar)

Adam optimizer

Optimizer (Caltech)

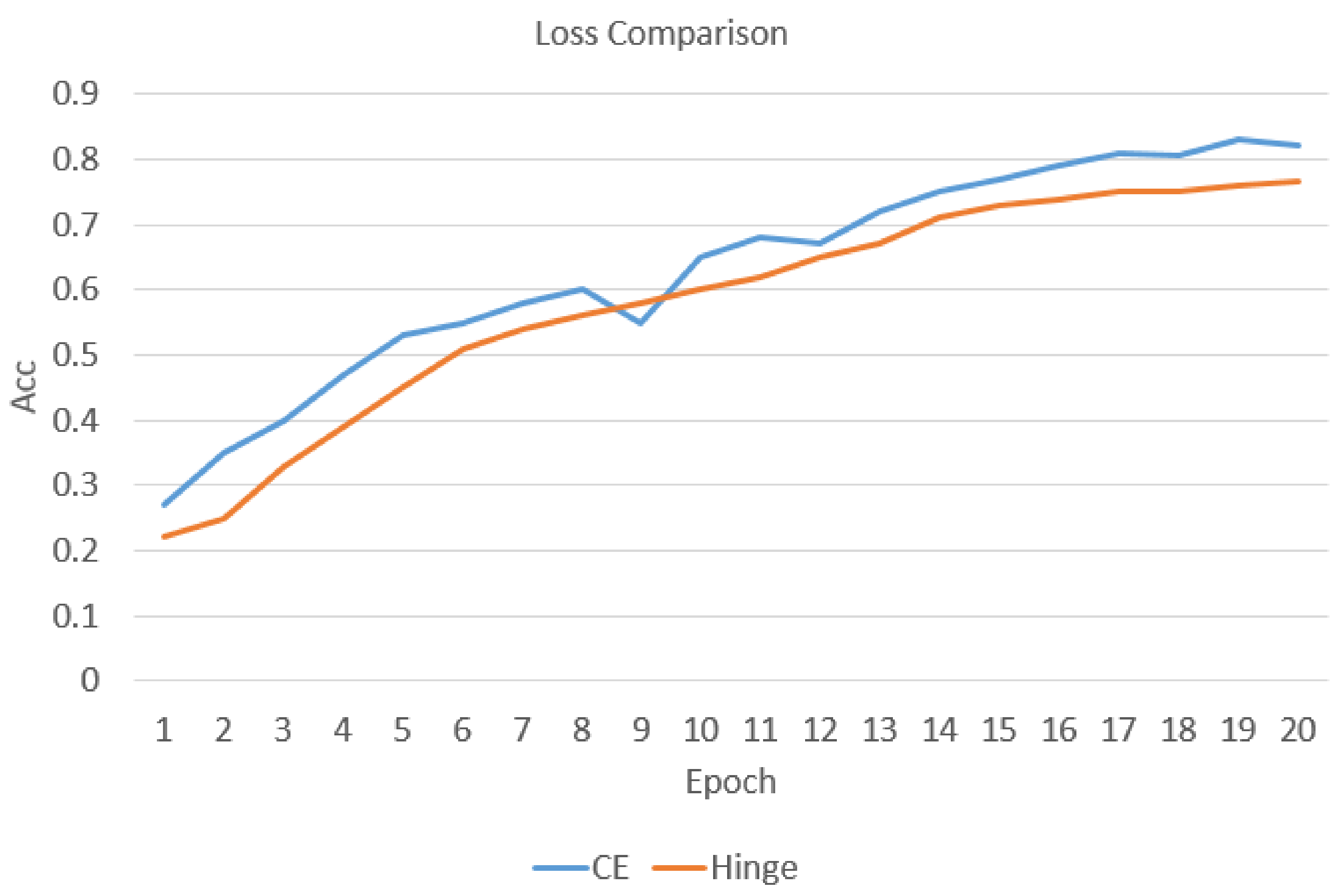
Adam vs SGD

Almost similar results



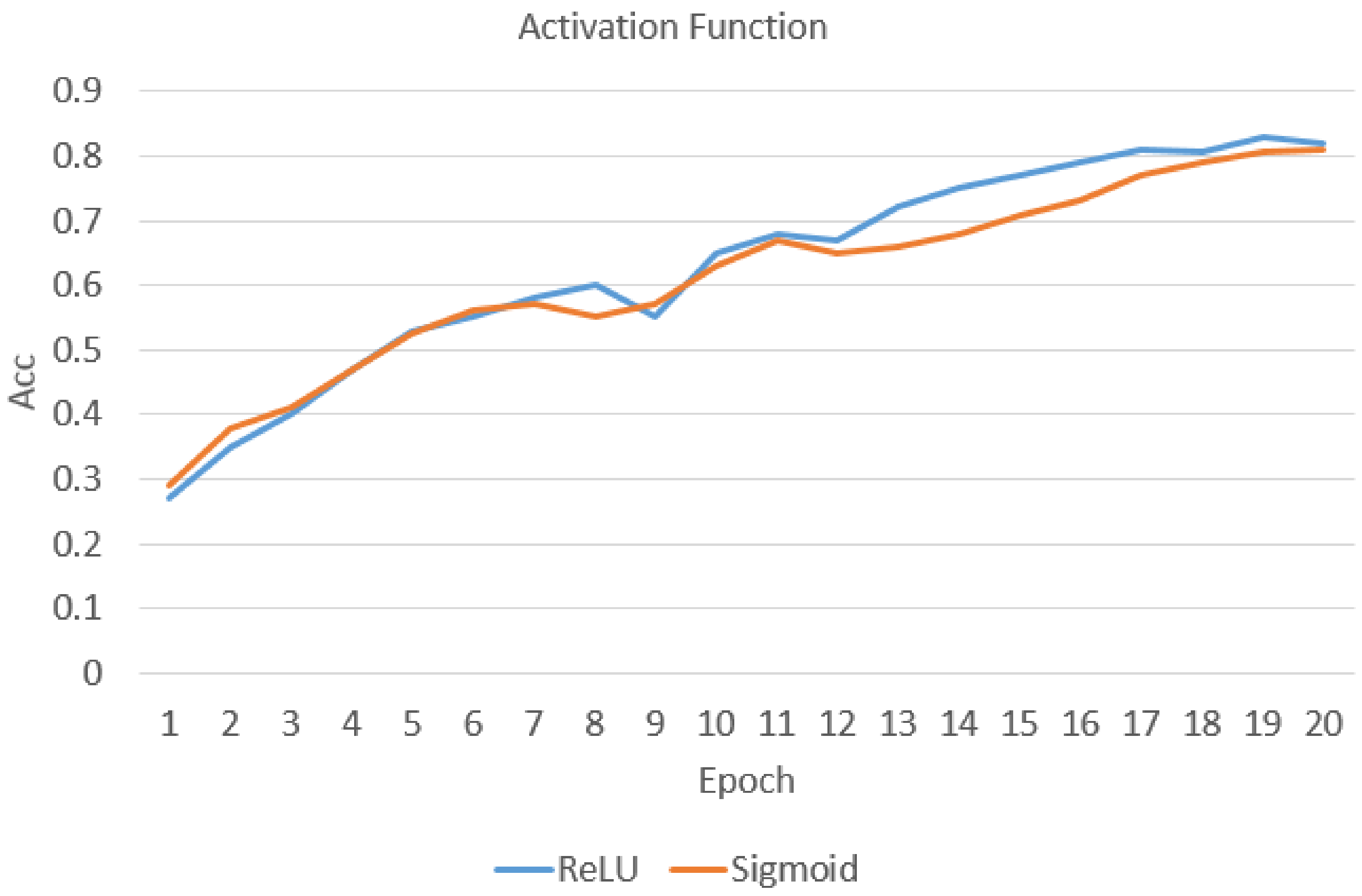
Loss Function (Caltech)

Accuracy using different loss functions



Activation Function (Caltech)

Similar results



Question?

https://openai.com/dall-e-3