Raunak Sarbajna, Md. Mahin and Christoph F. Eick

COSC 4368: Fundamentals of Artificial Intelligence Spring 2024

Problem Set 2 (Individual Tasks[[1]](#footnote-1) Centering on Neural Networks)

*Third Draft*

TASK 3: Build a Classical Neural Network Model

for the Fashion MNIST Dataset Raunak

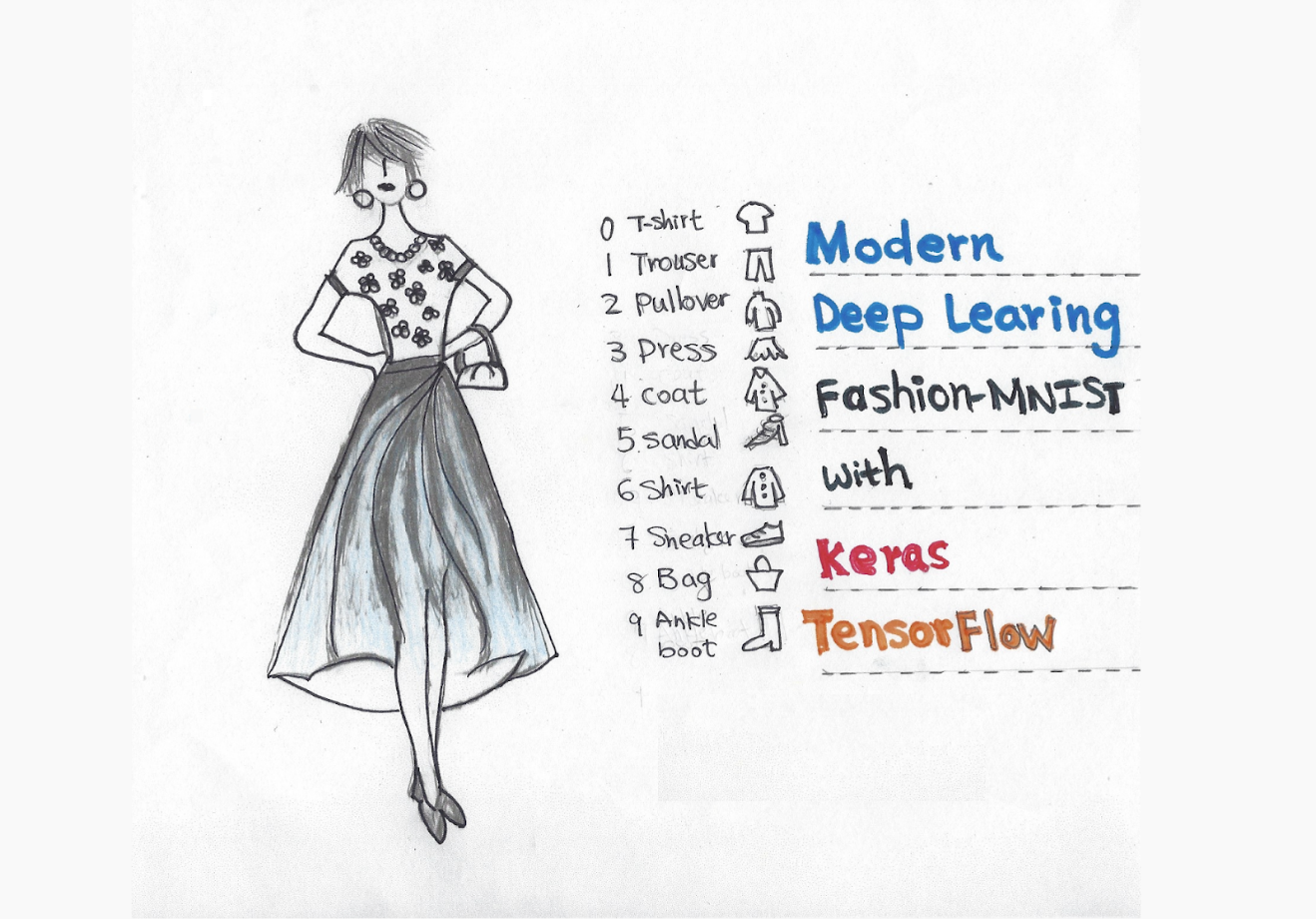


Fig. 1: Training a model to classify images is the archetypal neural network task

Submission Deadlines Task3: Tue., Mar 26, 11:59p;

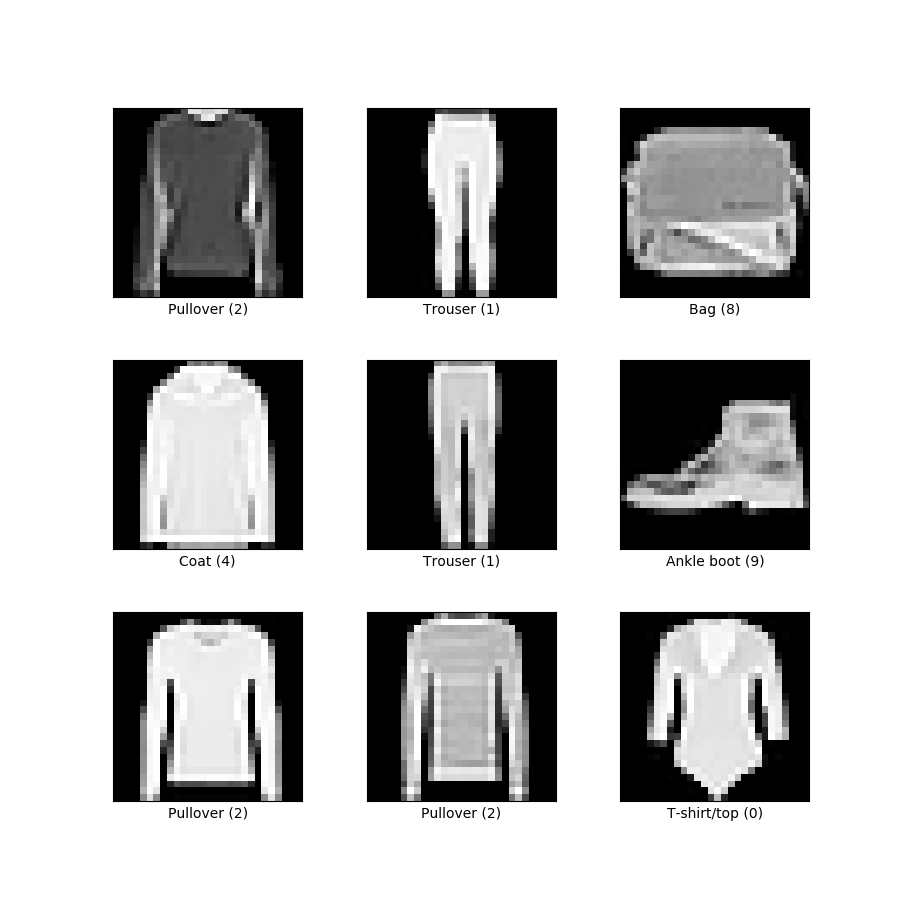
Last Updated: March 5, 10a.

Weight Task3: 25 points

In this task, you will explore the world of deep learning by building a neural network to classify clothing items in the Fashion MNIST dataset. You will learn how to build, train, and evaluate a model, gaining valuable insights into neural network capabilities.

**Dataset:**

Fashion MNIST is a popular image classification dataset containing 70,000 grayscale images of 10 different clothing items (T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot). We will use this dataset to train our neural network to identify these clothing items. Below some samples are given from Fashion MNIST dataset:



**Tasks:**

**1. Preprocessing** (You can use method used in notebook given for Task 4)

1. Download the Fashion MNIST dataset using TensorFlow or another suitable library.
2. Preprocess the data: normalize pixel values, split the data into training and testing sets.

**2. Building the Neural Network (5 pts)**

1. Choose a neural network architecture suitable for image classification. Consider using a Convolutional Neural Network (CNN) due to its effectiveness in this domain.
2. Define the network architecture in your chosen library (e.g., TensorFlow, PyTorch). Use layers like convolutional, pooling, and dense layers.
3. Compile the model, specifying the optimizer (e.g., Adam) and two separate loss function (e.g., categorical cross-entropy).

**3. Training and Evaluation (10 pts)**

1. Train the model on the training set with epochs numbering between 50 ~ 300.
2. Monitor the training process by logging metrics like accuracy and loss after each epoch.
3. Evaluate the model's performance on the test set using metrics like accuracy, precision, and recall for each class.
4. Experiment with different hyperparameters to improve the model's performance.
   * learning rate (0.1/0.25/0.30),
   * number of layers (2/3/5),
   * neurons per layer (10/20/30)

**4. Analysis (10 pts)**

1. Analyse the training and evaluation results. Are there any classes with particularly low accuracy?
2. Display the model's confusion matrix. Can you identify any interesting patterns or mistakes? Visualize the lowest performing class.
3. Determine which class is the most difficult to predict by seeing which has the lowest F2. Try to explain why this class is challenging to predict.

**Submission:**

* Submit a report including:
  + Clear explanation of your chosen architecture and its rationale.
  + Visualization of training and evaluation metrics (graphs, confusion matrices).
  + Your answers to the analysis questions.
  + Discuss the limitations of your model and potential improvements for future exploration.

**Bonus: (up to 5 pts)**

* Visual explanations concerning what kind of errors the Neural Network models makes and which pairs of classes are hard to distinguish for the Neural network model.
* Implement data augmentation techniques to improve model generalizability and accuracy.
* Apply your model to classify images from a different dataset of your choice.

**Resources:**

* Fashion MNIST dataset: <http://pytorch.org/vision/stable/generated/torchvision.datasets.FashionMNIST.html>
* TensorFlow tutorials: <https://www.tensorflow.org/tutorials>
* PyTorch tutorials: <https://pytorch.org/tutorials/>

**Pre-Processing Example:**

**A screen shot of a computer program

Description automatically generated**

**Task 4: Auto-Encoders** Mahin

Submission Deadline Task4: Sa. April 6, 11:59p;

Last Updated: March 31, 11a.

Weight: Task weight: 30 points

**Learning Objectives**:

1. Learn to use deep learning and generative models such as VAE
2. Learn to use classifiers
3. Learn differnt tools to create different deep learning models
4. Learning how to assess quality of models

|  |  |
| --- | --- |
| **A: Variational Autoencoder** | **B: Normal Autoencoder** |
| C:\Users\mdmah\Google Drive\UH\TA\AI\Spring 2023\Picktures\VAE.png | C:\Users\mdmah\Google Drive\UH\TA\AI\Spring 2023\Picktures\autoencoder_header.png |

Fig. 2.: Variational and Normal Autoencoder Architecture

**A**: Variational Auto-encoder Architecture: The VAE contains one encoder and one decoder part. Encoder starts from x,h and ends in z=(σ + μ). [(σ + μ) learns latent representation or key features of the images]. Decoder starts from z=(σ + μ) to h2 and ends in x2. Decoder utilizes learned important represntation from z=(σ + μ) and tries to regenerate the image in x2.

**B**: Noraml Auto-encoder Architecture: A normal autoenoder contains only a fully connected layer z instead of a pair of layers (σ + μ) to learn the hidden representation.

In this project we will use the Fashion MNIST computer vision digit dataset and experiment with auto-encoders such as Variational Auto-encoder(VAE) and simple autoencoder. The Jupiter notebook provided contains a VAE architecture and process of downloading the dataset. (total: 30 points)

**Task 4 Subtasks:**

1. Learn latent features from the Fashion MNIST dataset. Use the model given in reference [1]. Perform the following tasks: (total 10 points)
   1. The given model has a three layer architecture for each encoder and decoder part. Can you modify the architecture into a four layer format. In this task, you need to convert encoder part into (x, h1, h2, z=(σ + μ) )= (784\*400\*100\*20) and decoder part into (z=(σ + μ), h3, h4, x2 )= (20, 100, 400, 784). Finally you need to compare the results based on their:
      1. Optimal loss after the model is fully trained, and
      2. Visually inspecting the output they generate using the images they generate and reconstruct. You can use plot\_generation() and plot\_reconstruction() function from the notebook.

Based on optimal loss and visual inspection write down your opinion which model is better and also try to give an explanation why a model is giving good performance over another.

* 1. Take the base three layer architecture and check the performance of the model for six different configuration, where h\_dim and z\_dim is changed into following patterns: [(400,50), (400, 10), (400, 30), (300, 35), (300, 5), (300,40)](Note: First one is the base architecture). Perform the same type of comparison you have done in task **a** using optimal loss of the model and visual inspection and write down your opinion which model is better and also try to give an explanation why a model is giving good performance over another. (total 10 points)
  2. Take the best architecture from b and convert it into a normal auto-encoder (figure 1.b)[2], e.g. replace z\_dim such a way that it will be single layer. As noraml autoencoder and variational auto-encoder have very different way of loss calculation, you need to modify loss function too. Now Perform the same type of comparison you have done in task **a** using optimal loss of the model and visual inspection and down your opinion which model is better and also try to give an explanation why a model is giving good performance over another. (total 10 points)

## **Deliverables:**

1. A Jupyter notebook with your code and analysis. Your notebook should use markdown and should contains:
   1. Description of the code or changes you made to the code for every task in the markdown (before each code section, also comment properly within the code) (code + description 5 points for each task)
   2. You should describe the loss comparison using markdown after each task, e.g. try to answer the task a, b, c using markdown in the notebook after completing each task (2 points for each task)
   3. Similar to b, try to explain visual comparisons using markdown after each task, e.g. try to answer the task a, b, c using markdown in the notebook after completing each task (2 points for each task)

1. A report that will be pdf generated from the markdown. But remember to do following changes (1 points for each task)
   1. Add discussion of tasks you performed but do not include code in the report
   2. All your comparison description. Remember to add the model outputs before each comparison.

**References:**

1. <https://github.com/dataflowr/notebooks/blob/master/HW3/VAE_clustering_empty.ipynb>
2. <https://www.analyticsvidhya.com/blog/2021/06/complete-guide-on-how-to-use-autoencoders-in-python/>

**Task 5: Learning and Using Diffusion Models** Raunak

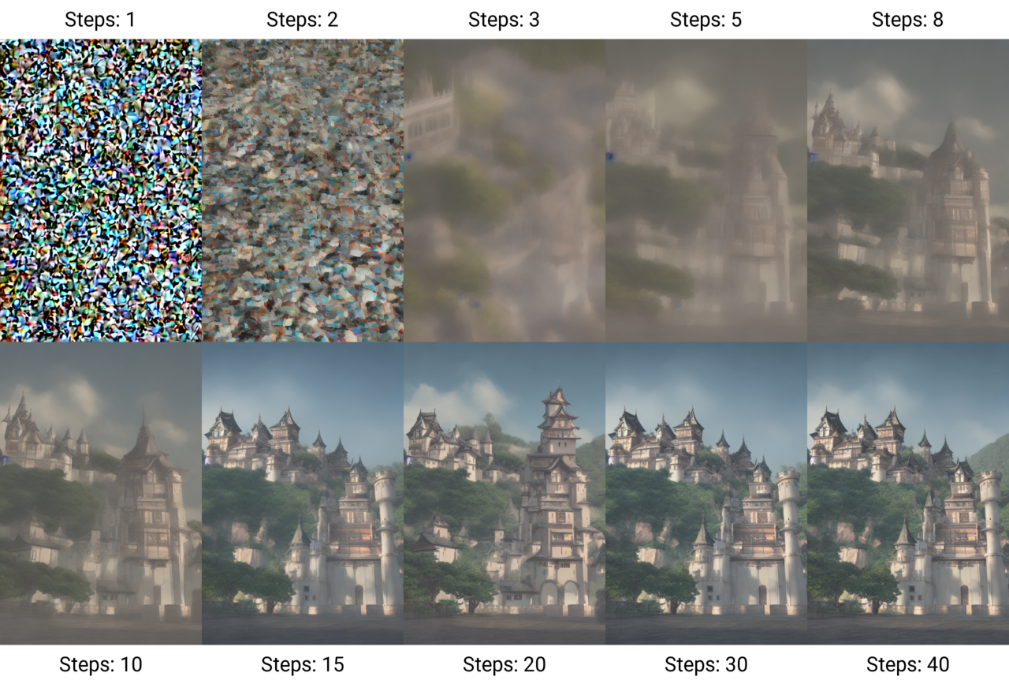


Fig. 3: The Denoising Process used by *Stable Diffusion*

Submission Deadline Task5: Thursday, April 26, 11:59p

Last Updated: April 12, 8a

Task Weight: 35 points

This assignment is a short exploration of a basic diffusion model. You'll be working with a pre-trained diffusion model in a Jupyter notebook environment to generate images and analyze how different settings influence the results.

Link: <https://github.com/RaunakDune/BasicDiffusionModel/>

**Learning Objectives:**

1. Gain hands-on experience with diffusion models using a Jupyter notebook.
2. Experiment with tuning hyperparameters in a pre-trained diffusion model.
3. Analyze the generated images and interpret the impact of parameter changes.
4. Effectively document and visualize your findings within the notebook.

**Dataset**

**Link:** [**https://drive.google.com/file/d/1NF-JNSVkdu\_lG7UVVtTAUMWIRQANqtav/view?usp=sharing**](https://drive.google.com/file/d/1NF-JNSVkdu_lG7UVVtTAUMWIRQANqtav/view?usp=sharing)

The Stanford Cars dataset is a widely used benchmark dataset in the field of computer vision and machine learning, particularly for tasks related to object recognition and classification. It is curated and maintained by researchers at Stanford University.

The Stanford Cars dataset consists of images of cars belonging to 196 classes, where each class represents a different car model. The dataset is further divided into training and test sets, with a total of 16,185 images in the training set and 8,054 images in the test set. Each image in the dataset is annotated with bounding boxes around the car, along with additional metadata such as the make, model, and year of the car. This annotation facilitates fine-grained classification tasks, where the goal is to classify images into specific car models. However, we will *not* be using these annotations for this task.

**Task 5 Subtasks**

* **Understanding the Existing Diffusion Model (No Points)**
  1. Review the provided Jupyter Notebook file containing an existing diffusion model implementation.
  2. Familiarize yourself with the structure of the code and understand how the diffusion process is simulated.
  3. Identify key parameters in the diffusion model, such as the diffusion rate, initial conditions, and time steps.
  4. Run the model once from the start to get a benchmark of the result and the time taken.
* **Implementing Parameter Tuning for Forward Diffusion (10 Points)**
  1. Explain what these parameters from the existing diffusion model do: T, IMG\_SIZE, BATCH\_SIZE.
  2. Create a new code cell in the Jupyter Notebook to modify the selected parameters.
  3. Conduct a series of simulations by varying the values of the chosen parameters while keeping other parameters constant:
     1. T: 250/300/350
     2. IMG\_SIZE: 16/32/64/128
     3. BATCH\_SIZE: 64/128/256
  4. Record the results of each simulation, including the quality of image and time taken for diffusion.
* **Modify the Backward Process and Loss: (10 Points)**
  1. Explore the U-Net implementation and understand what it’s doing. Write a 1 to 3 line summary.
  2. Implement two different activation functions of your choice. Some suggestions are present in the notebook. Explain your choice.
  3. Explore two different loss functions of your choice in addition to the one present in code. Explain your choice.
* **Training and Experimentation (10 Points)**
  1. Choose the fastest and second fastest parameter values from **Question 2** for your training.
  2. Explore the results for each of the activation functions and loss functions.
  3. Create a variable to track the loss values and plot it.
  4. Observe how the generated cars look. Why do they look so poor? How does the quality change with IMG\_SIZE?
* **Analysis (5 Points)**
  1. Summarize your findings from all the experiments.
  2. Discuss any trends or patterns observed in the results.
  3. Provide insights into how each parameter affects the rate and extent of diffusion.
  4. Reflect on the strengths and limitations of the existing diffusion model.

**Deliverables**

* Save your modified Jupyter Notebook with your changes and analysis.
* Ensure that your Notebook includes clear explanations, code comments, and visualizations.
* *Submit your Notebook file (.ipynb)* along with any additional resources (e.g., data files) to Teams. **Don’t zip your submission.**
* Make sure to adhere to academic integrity guidelines and cite any external sources used in your assignment.

**References:**

* **Jupyter Notebook:** [**https://github.com/RaunakDune/BasicDiffusionModel/**](https://github.com/RaunakDune/BasicDiffusionModel/)
* **Dataset:** [**https://drive.google.com/file/d/1NF-JNSVkdu\_lG7UVVtTAUMWIRQANqtav/view?usp=sharing**](https://drive.google.com/file/d/1NF-JNSVkdu_lG7UVVtTAUMWIRQANqtav/view?usp=sharing)

1. Collaboration with other students is not allowed! [↑](#footnote-ref-1)