HW 5: Deploying Machine Learning Models
CPE 490/590 ST

Instructor: Rahul Bhadani

Due: April 13, 2025, 11:59 PM
155 points

You are allowed to use a generative model-based AI tool for your assignment. However, you must submit an accompanying reflection report detailing how you used the AI tool, the specific query you made, and how it improved your understanding of the subject. You are also required to submit screenshots of your conversation with any large language model (LLM) or equivalent conversational AI, clearly showing the prompts and your login avatar. Some conversational AIs provide a way to share a conversation link, and such a link is desirable for authenticity. Failure to do so may result in actions taken in compliance with the plagiarism policy.

Additionally, you must include your thoughts on how you would approach the assignment if such a tool were not available. Failure to provide a reflection report for every assignment where an AI tool is used may result in a penalty, and subsequent actions will be taken in line with the plagiarism policy.

Submission instruction:

Submission instruction for this homework supersedes one mentioned in the Syllabus.

This homework requires all answers recorded in a single `.ipynb` Python notebook. You may upload hand-written PDF for the theory portion. You can use a combination of text cell (i.e. markdown formatted cell) and code cell to provide your answer. To add equations you should be able to use Latex syntax in the text cells of your Python notebook. As a part of your submission, you must provide executed notebook with code, text, and outputs. Alternatively, you can also provide a url (whose permission you must change to 'anyone with link can view') of your Python notebook from Google Colab. The naming convention for your notebook should follow the format {firstname_lastname}_CPE 490/590 ST_hw03.ipynb. For example, if your name is Sam Wells, and you are enrolled in CPE 490 your file name should be sam_wells_CPE490_hw03.ipynb.

Please refer to https://github.com/rahulbhadani/CPE490_590_Sp2025/blob/master/Code/ for hands-on.

Please refer to https://github.com/rahulbhadani/iris_classifier_app for an example web app to make an inference on IRIS dataset.

Practice

1 Machine Learning Deployment for MNIST Dataset

1.1 Convolutional Neural Network for MNIST Dataset

Create a Convolutional Neural Network for MNISIT digit classification.

1. Normalize the dataset so that pixel values are between 0 and 1. **(5 Points)**.
2. Divide the dataset into training and test dataset such 1/3rd of the dataset is set aside for the test dataset. **(5 Points)**.
3. Implement CNN model with the given Architecture specification **(25 Points)**:

Table 1: CNN Architecture Specification

Layer Type	**Parameters**	**Output Shape**
Input	–	(28, 28, 1)
Conv2D	Filters=32, Kernel=(3,3), ReLU	(26, 26, 32)
MaxPooling2D	Pool size=(2,2)	(13, 13, 32)
Conv2D	Filters=64, Kernel=(3,3), ReLU	(11, 11, 64)
MaxPooling2D	Pool size=(2,2)	(5, 5, 64)
Flatten	–	(1600)
Dense (Fully Connected)	Units=128, ReLU	(128)
Dropout	Rate=0.5	(128)
Dense (Output)	Units=10, Softmax	(10)

4. Choose appropriate training parameters such as learning rate, optimizers, etc. **(5 Points)**.
5. Bonus **(15 Points)**: Implement Learning Rate Scheduler
6. Save the model as ONNX format along with randomly drawn five samples. **(10 Points)**.
7. Load the ONNX model in your Jupyter Notebook, and test whether your model correctly predicts the digit images. **(10 Points)**.

1.2 Developing Flask based Web Applications

1. Load the pre-trained ONNX model using onnxruntime in Python. Write the predict function in Python that takes 28×28 grayscale image as input. Normalize it so that pixel values from 0 to 1. The function should make inference and return the predicted digit with confidence score (a set of ten numbers representing probability of belong to each digit class). **(10 Points)**.

2. Backend: Create a webserver with two routes **(25 Points)**:

   (a) / (Homepage): Renders an HTML form for image upload.

   (b) `/predict` (POST endpoint) that accepts image uploaded, if image is not 28, it resizes to 28 and convert to grayscale, Calls the ONNX prediction function written in the previous part, and returns prediction results (digit + confidence) as JSON or rendered HTML.

3. Front end **(25 Points)**:

   - Create a simple HTML form with that lets user upload an image file.
   - Upon Uploading, image file should be displayed on the web page
   - A predict button should be there that upon clicking submits the image to the server.
   - A display area tha upon submitting should display the correctly predicted digit along with chart.js enable bar plot showing probabilities of each digit class.
   - A sample area that should show some example sample MNIST figures that users may select to make prediction if the users don't have their own image files.

4. Error handling **(15 Points)**:

   (a) If the uploaded file is not an image, client side scripting should raise an error and do not submit the file to the server.

   (b) If no file is uploaded, the predict button should not do anything.

5. README **(10 Points)**:

   (a) Provide a READM.md file that explains how to run your web app.

6. Github: Create a Github repository and upload your Flask Web App to the Github along with the README.md, and ONNX model. You may choose to upload Python notebook using which you created a trained ONNX model in the same GitHub repository and provide its URL as a part of the submission. **(10 Points)**.