Identifying Appliances from Energy Use Spectrograms

IBM Advanced Data Science Capstone

George Pinto

The Dataset

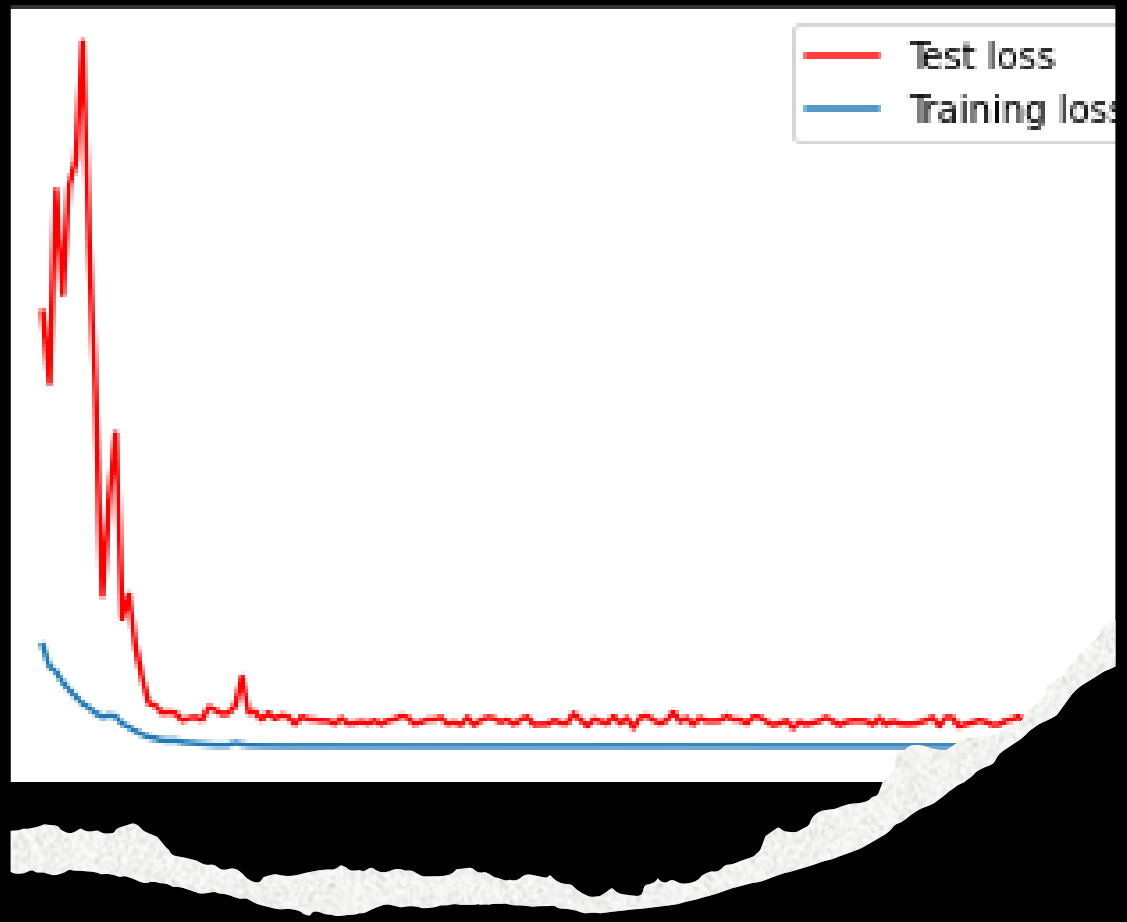
- Spectrogram images
- 1152 training images and 768 testing images sized 128 x 176 pixels
- This plug load dataset contains current and voltage measurements sampled at 30 kHz from 11 different appliance types present in more than 60 households in Pittsburgh, Pennsylvania. Plug load refers to the energy used by products that are powered by means of an ordinary AC plug (i.e., plugged into an outlet).
- For each appliance, plug load measurements were post-processed to extract a three-second-long window of measurements of current and voltage. For some observations, the window contains both the startup transient state (turning the appliance on) as well as the steady-state operation (once the appliance is running). For others, the window only contains the steady-state operation. The observations were then transformed into two spectrograms, one for current, and one for voltage.

Use Case

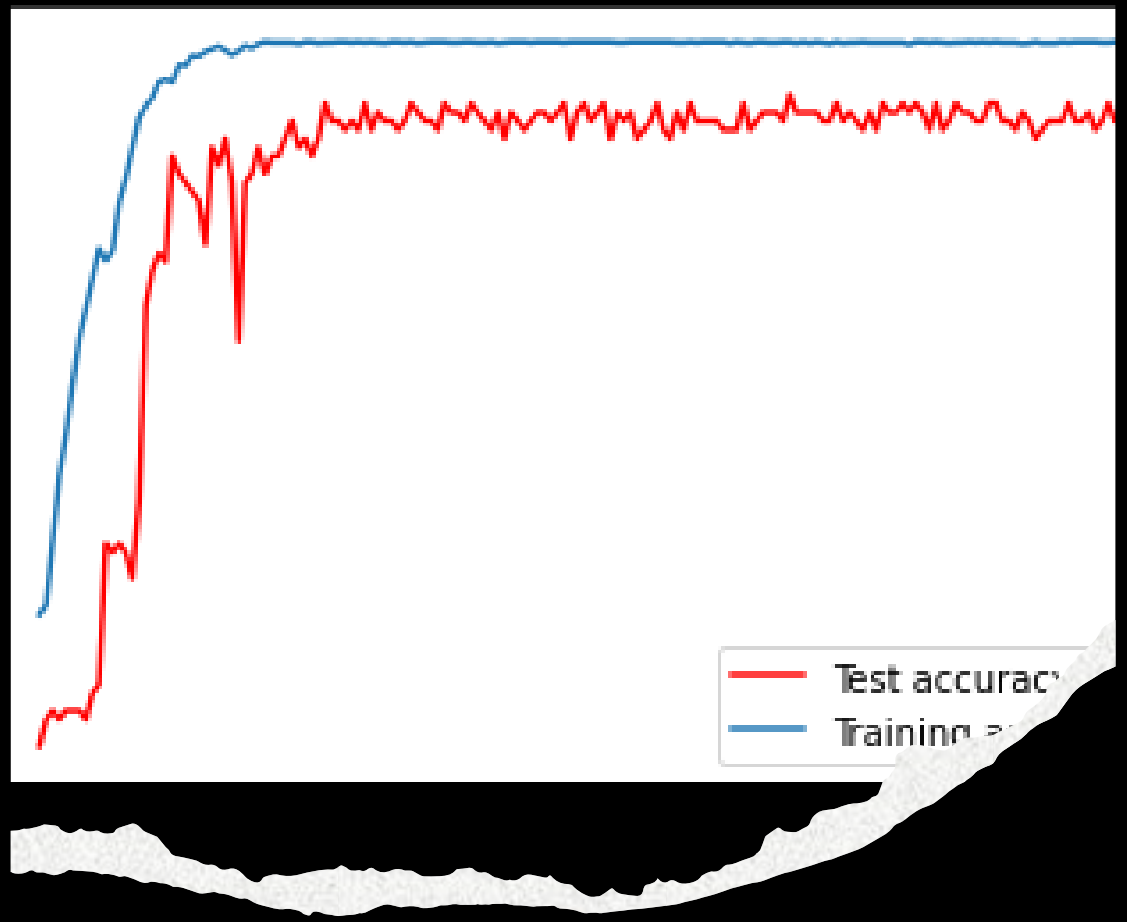
- U.S. Energy Information Administration projects a 28% increase in world energy consumption by 2040. And the energy sector is a major contributor to climate change. Energy production and use accounts for more than 84% of U.S. greenhouse gas emissions.
- Increasing the efficiency of energy consumption has benefits for consumers, providers, and the environment. With an increasing number of IoT devices coming online in the energy sector, there is more and more data that can be used to monitor and track energy consumption. Ultimately, this type of data can be used to provide consumers and businesses with recommendations on ways to save energy, lower costs, and help the planet.
- Use standard AI tools to identify 11 different types of appliances from their electric signatures, quantified by current and voltage measurements.


conv_1_input: InputLayer | input: [(?, 128, 176, 1)] | output: [(?, 128, 176, 1)]
conv_1: Conv2D | input: (?, 128, 176, 1) | output: (?, 128, 176, 32)
batch_normalization: BatchNormalization | input: (?, 128, 176, 32) | output: (?, 128, 176, 32)
pool_1: MaxPooling2D | input: (?, 128, 176, 32) | output: (?, 64, 88, 32)
dropout: Dropout | input: (?, 64, 88, 32) | output: (?, 64, 88, 32)
conv_2: Conv2D | input: (?, 64, 88, 32) | output: (?, 64, 88, 64)
batch_normalization_1: BatchNormalization | input: (?, 64, 88, 64) | output: (?, 64, 88, 64)
pool_2: MaxPooling2D | input: (?, 64, 88, 64) | output: (?, 32, 44, 64)
dropout_1: Dropout | input: (?, 32, 44, 64) | output: (?, 32, 44, 64)
conv_3: Conv2D | input: (?, 32, 44, 64) | output: (?, 32, 44, 128)
batch_normalization_2: BatchNormalization | input: (?, 32, 44, 128) | output: (?, 32, 44, 128)
pool_3: MaxPooling2D | input: (?, 32, 44, 128) | output: (?, 16, 22, 128)
dropout_2: Dropout | input: (?, 16, 22, 128) | output: (?, 16, 22, 128)
conv_4: Conv2D | input: (?, 16, 22, 128) | output: (?, 16, 22, 256)
batch_normalization_3: BatchNormalization | input: (?, 16, 22, 256) | output: (?, 16, 22, 256)
pool_4: MaxPooling2D | input: (?, 16, 22, 256) | output: (?, 8, 11, 256)
dropout_3: Dropout | input: (?, 8, 11, 256) | output: (?, 8, 11, 256)
conv_5_input: InputLayer
conv_5: Conv2D
batch_normalization_4: BatchNormalization
pool_5: MaxPooling2D
dropout_4: Dropout
conv_6: Conv2D
batch_normalization_5: BatchNormalization
pool_6: MaxPooling2D
dropout_5: Dropout
conv_7: Conv2D
batch_normalization_6: BatchNormalization
pool_7: MaxPooling2D
dropout_6: Dropout
conv_8: Conv2D
batch_normalization_7: BatchNormalization
pool_8: MaxPooling2D
dropout_7: Dropout
add: Add | input: [(?, 8, 11, 256), (?, 8, 11, 256)] | output: (?, 8, 11, 256)
Flatten: Flatten | input: (?, 8, 11, 256) | output: (?, 22528)
dense_1: Dense | input: (?, 22528) | output: (?, 1024)
batch_normalization_8: BatchNormalization | input: (?, 1024) | output: (?, 1024)
dropout_8: Dropout | input: (?, 1024)

Model



Results - Validation Loss 0.5573



Results – Validation Accuracy 92.5

Advantages

- Dual input model can be fed current images (input1) and voltage images (input2) by easily uploading spectrogram images on the web (can be deployed using Tensorflow serving.
- Constructed using open-source data (Jupyter Notebooks, Pandas, TensorFlow2), built and stored in the cloud (Google Colab using GPUs, stored on Google Drive). Pushed to repository on GitHub for version and license control.
- Scalable – built using TensorFlow 2 using generators which means the model can be fed in batches when needed to classify large amounts of data
- 92.5% accuracy on test data (identifies 11 categories of appliances based on the spectrogram images) – the accuracy should improve if more data is used for training