Capstone 2019:
Neural Architecture Search

Data Exploration & Preliminary Results

Graphene Kirigami Dataset Overview

The graphene kirigami data set describes the mechanical properties (stress and strain) of an extremely thin slice of graphene that is just one molecule thick.

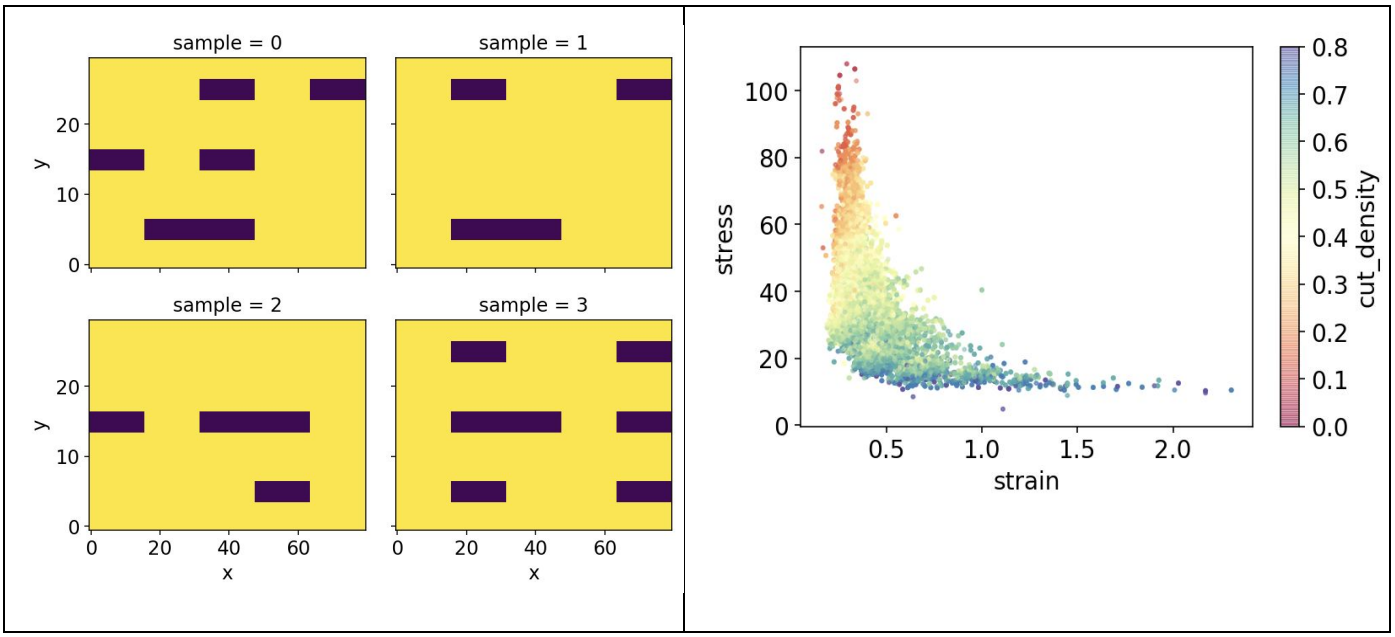

Figure 1a: Example cuts of graphene. **Figure 1b:** Mechanical properties of cuts

A 3x5 pattern of cuts is tiled onto a 30x80 grid, whose properties are then simulated using the LAMPPS molecular simulator developed at Sandia labs. The above figures demonstrate sample cuts, and provide an overview of the mechanical properties of the ~30,000 samples that were simulated.

Neural Architecture Search on Graphene Kirigami Data

The machine learning task is to predict the mechanical properties given the 3x5 cut pattern (coarse grid) or 30x80 pattern (fine grid). It seems that this is a relatively easy task. A simple random forest (trained and tested on the coarse grain dataset) performed on par with more sophisticated models. On the more complex, fine-grid dataset, we ran DARTS and the resulting model achieved a test R^2 of 0.9225, which is similar to the best results obtained with other methods (including those in the original Graphene Kirigami publication). Below are

visualizations of the normal (stride 1) and reduction (stride 2) cells discovered during the architecture search on the graphene data set.

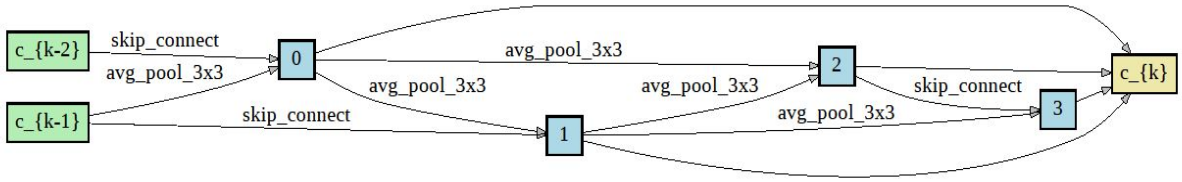

Figure 2: Learned Cell for Graphene (Normal)

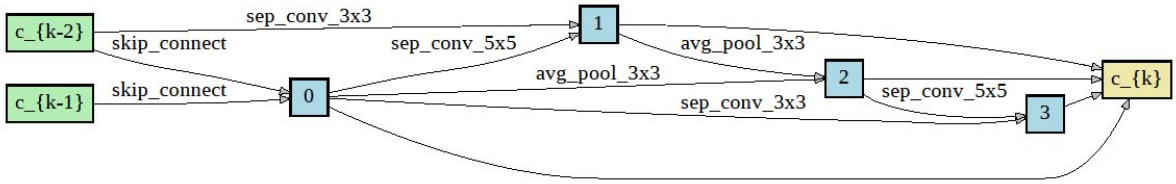

Figure 3: Learned Cell for Graphene (Reduce)

In visualizing these cells, we see that many of the operations are skip connections, average pooling, or max pooling, none of which have any trainable parameters. This is intuitive and reflects that the problem is “too easy” for a model of this size.

We ran into some difficulties when we used parameters developed for CIFAR-10 on the graphene data. We ended up training a shallower network, with only 8 total layers rather than the default of 20 layers. The learning curve below shows us that the model trained well. There is an unusual behavior, though, in which it performs worse in training than on validation / test, and the training performance starts to decay after ~75 epochs. We verified that this was a real effect and not an error reading the log files. Our intuition is that the model has too many parameters and is changing weights too fast on the training samples, but in a way that is not adversely impacting the validation / test samples. The table at bottom summarizes the performance, parameter size, and training times of three architectures used on the graphene data.

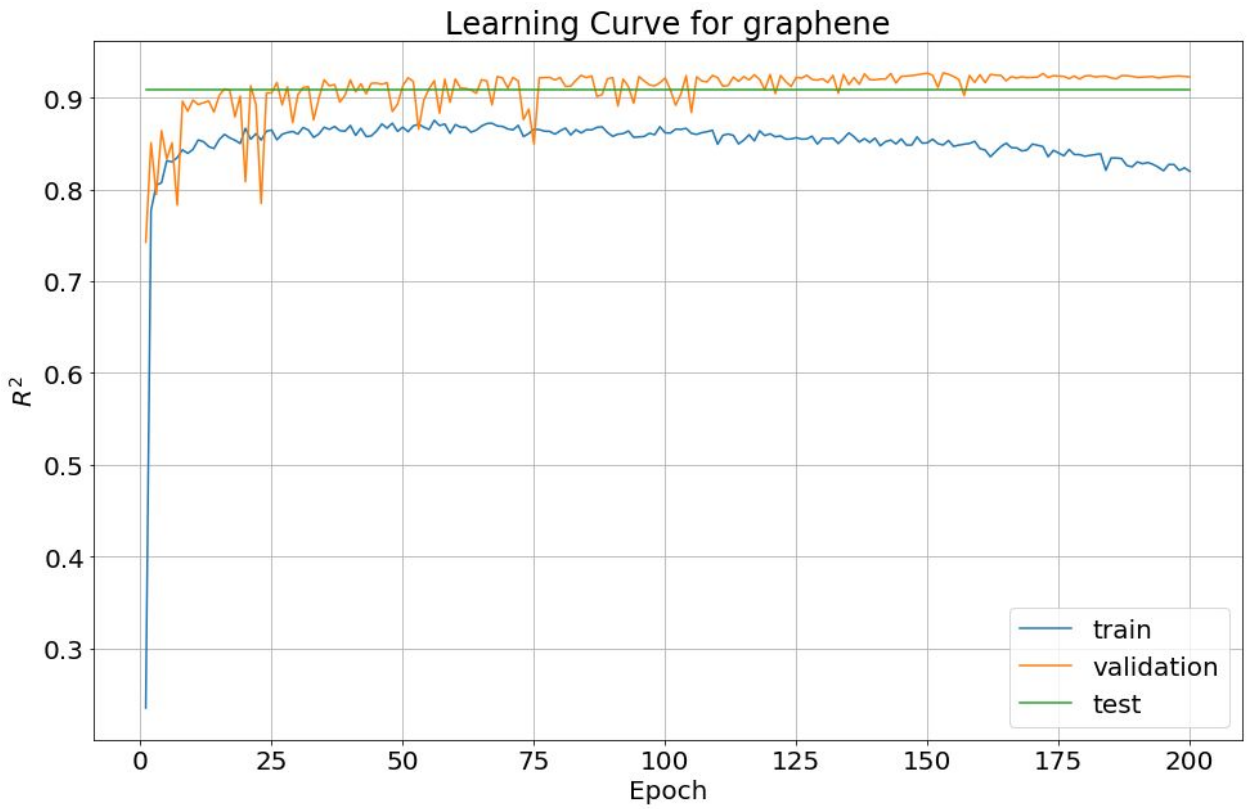

Figure 4: Learning curve for Graphene dataset (DARTS)

Method / model	**No. Parameters**	**Training time[1] (minutes)**	**Test R^2**
DARTS	195,236	1042	0.92
ResNet-18	11,168,193	11	0.92
"Tiny ResNet" (10 layers, 8x less filters)	77,273	4	0.92

Table 1: Summary of results of DARTS and hand-designed (ResNet) architecture on Graphene dataset

[1] Time reported to train for 30 epochs on a single GPU.

Neural Architecture Search on CIFAR-10

The DARTS paper focused on two classic ML tasks: image classification and language modeling. In image classification, their benchmark task was to perform an architecture search on CIFAR-10, then train the identified architecture to convergence. We assume that our audience consists of domain experts, so we will not belabor an explanation of the CIFAR-10 data set. Suffice it to say that it is a standard image classification data set with 32x32 pixel low resolution images of objects in 10 categories that could easily have come out of a children's book.

We began by replicating the CIFAR-10 training. This made us familiar with their code and also established a baseline. We easily achieved a test accuracy of 97.10% on the most likely guess, a respectable result in line with the DARTS authors. The learning curve below shows that training proceeded smoothly as one would hope.

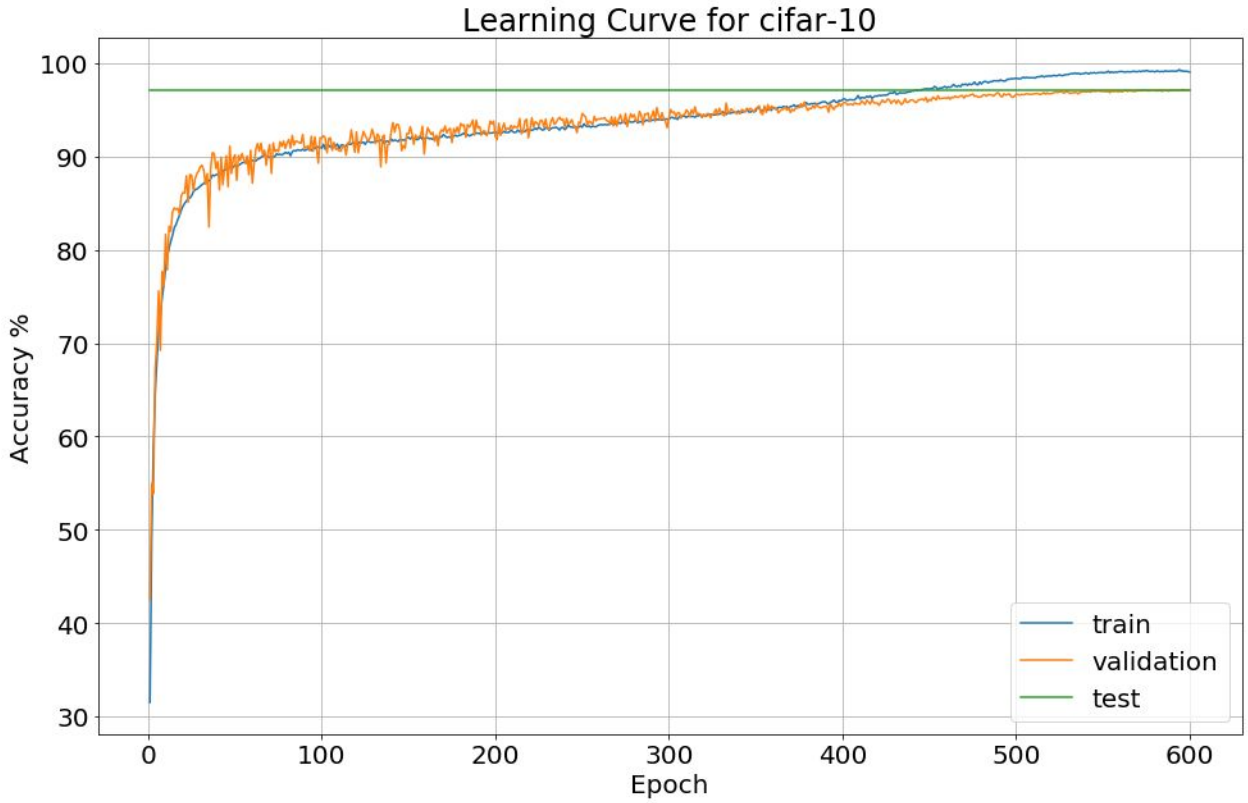

Figure 5: Learning curve for CIFAR-10 dataset (DARTS)

Sparsity Pattern of Learned Weights: Graphene vs. CIFAR-10

After we trained the model, we also wanted to explore the sparsity pattern of the learned edge weights. One of our research questions is whether NAS will learn sparse or dense networks. Here we saw an interesting distinction between the graphene and CIFAR-10 data sets.

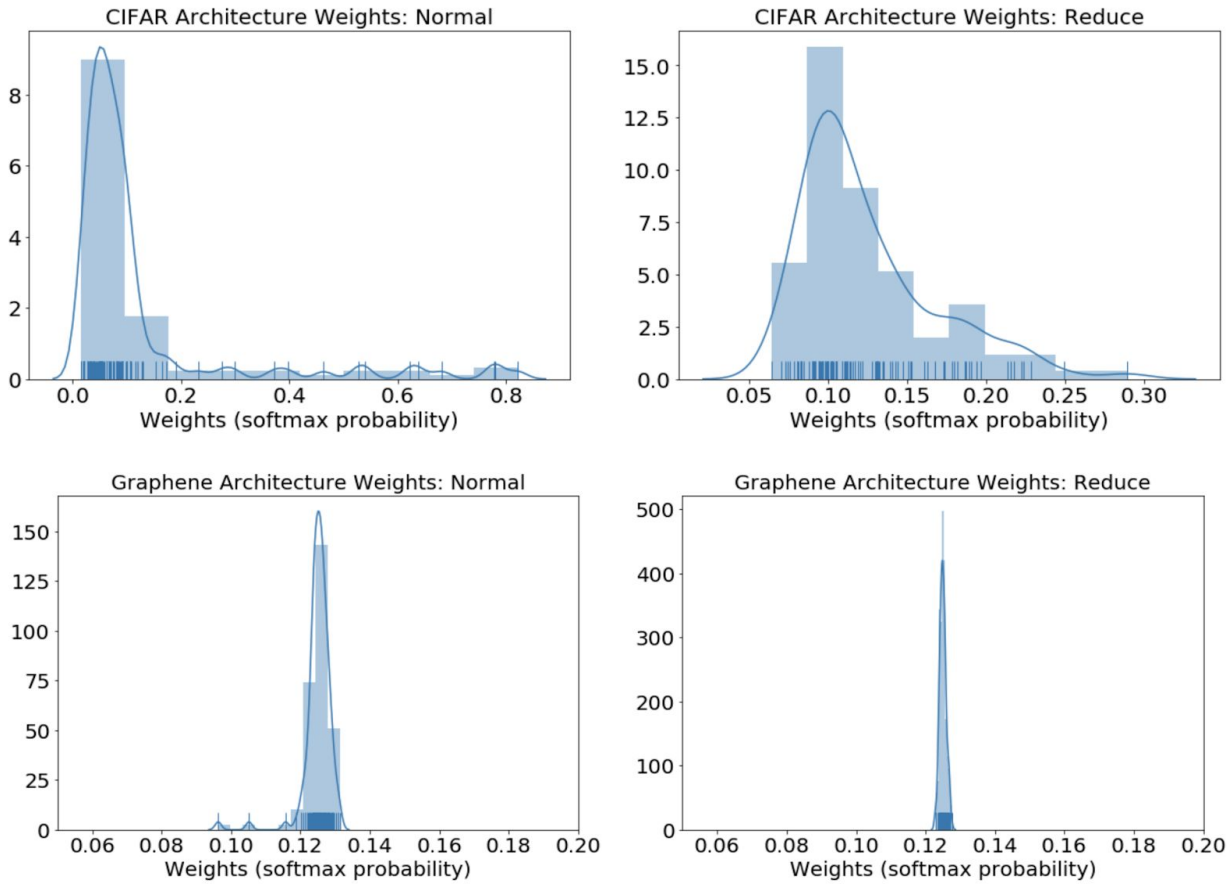

Figure 5: Distribution of weights (softmax probabilities) for DARTS architecture on CIFAR and Graphene datasets.

The weights on the cells learned for CIFAR-10 during the architecture search showed a noticeable spread over time. The search was learning that some operations work better than others. By comparison, the weights learned on the graphene cells remained closely clustered around their starting value of 0.125 (there are 8 possible operations in this search, and the initial weights are all close to ⅛).

Links to Code and Jupyter Notebooks with EDA

See our updated DARTS code here:
https://github.com/capstone2019-neuralsearch/darts

See our full repo, containing a superset of the results shown here:
https://github.com/capstone2019-neuralsearch/AC297r_2019_NAS

See here for more details on Graphene Kirigami, specifically:
https://github.com/capstone2019-neuralsearch/AC297r_2019_NAS/tree/master/notebooks/data_exploration/graphene_kirigami