1 APPENDIX

1.1 IMPLEMENTATION ENVIRONMENT DETAILS

For LSTM and Transformer, we used the PyTorch **?** package. For CNN, we used the Tensorflow **?** package to build the AL models (vanilla CNN and a visual geometry group (VGG) network (**?**)). We trained the models with GeForce 3090 GPUs on Ubuntu Linux 20.04. Since current hardware development does not support actual pipeline training, we simulated pipeline training by blocking gradient flows between AL layers with `Tensor.detach()` in PyTorch and with `tf.stop_gradient()` in TensorFlow.

1.2 TEXT CLASSIFICATION

During training, the most frequent 30k words in the training corpus were included in the word embedding. We replaced the remaining words with the [UNK] token. We trained all text classification experiments with 50 epochs. We used the Adam optimizer.

1.2.1 BI-LSTM

For Bi-LSTM, we applied gradient clipping with the value 5. For the standard version of Bi-LSTM, we experimented with learning rates of 0.0005 and 0.001, and report that with better accuracy: we used 0.0005 for AG's news and IMDB, and 0.001 for DBpedia and SST. We also adjusted the learning rate for AL, but the learning rate has little influence on the accuracy and the convergence speed. We used a batch size of 64 for all Bi-LSTM experiments. For IMDB and DBpedia, we set the max sequence length to 500. For AG's News and SST, we set the max sequence length to the longest sequence length; the exact length settings were applied to the Transformer experiments. The objective of the standard Bi-LSTM was cross-entropy loss. Details are given in Table 1.

Table 1: Bi-LSTM settings.

	emb dim	lstm1 dim	lstm2 dim	g dim	h dim	lr
LSTM	300	350	350	–	–	0.0005
LSTM-AL	300	300	300	128	128	0.001

1.2.2 TRANSFORMER

For the standard Transformer model, we used negative log-likelihood as the objective with multi-head attention (with 6 heads). We set the batch size to 128 because the standard Transformer converges only with a large batch size in our experiment. See Table 2.

Table 2: Transformer settings.

	emb dim	layer1 dim	layer2 dim	head	g dim	h dim	lr
Tran	300	512	512	6	–	–	0.00025
Tran-AL	300	256	256	6	128	128	0.00025

1.3 IMAGE CLASSIFICATION

We normalized the image pixels for both FashionMNIST and CIFAR-10 datasets as a preprocessing step. Since CIFAR-10 is more challenging, we also performed data augmentation on this dataset by resizing, random cropping, random flipping, and adjusting the brightness of the images. We set the batch size to 128. We set the initial learning rate to 0.0001 and applied scheduled decay until converging. We used Adam as the optimizer and cross-entropy loss for the standard version.

Vanilla CNN Model We implemented the vanilla CNN by modifying the previous code. The CNN contains 13 hidden layers and an output layer. The first 8 hidden layers are convolution layers; the last is flattened to a fully connected layer with 1280 neurons. The subsequent layers consist of 4

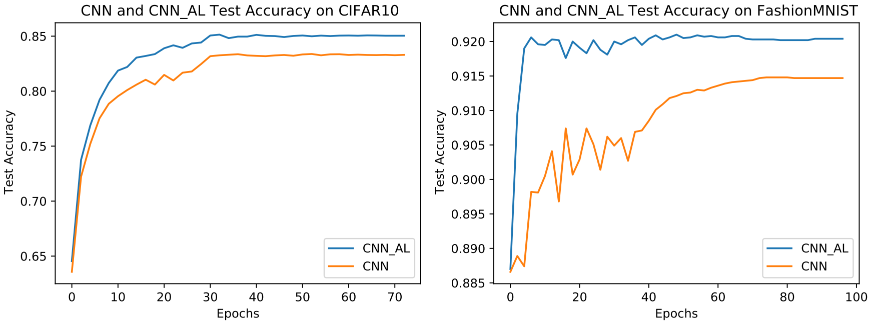

Figure 1: Testing accuracy vs. epochs for CNN and CNN-AL on CIFAR-10 and FashionMNIST.

feedforward layers, followed by an output layer with 10 classes. We set the initial learning rate to 0.0001 and applied scheduled decay.