AUTOKERAS

INDEX

1. Custom layers created for use of Autokeras framework for Lattice
   1. Convolution block
   2. Classification Head
2. Models trained so far
3. Environment to train the network
4. How to train the model using provided script
5. Limitations/Restrictions

1. CUSTOM LAYERS :

We can create custom layers in Autokeras as per our requirement. For Lattice NNC , we needed Batch-normalization, 8-bit kernel quantization and 8-bit activation quantization for crosslink device. So we created custom layer for Convolution and Dense layer. Also each Convolution layer is customized to have CBSR structure and an alteranate MaxPooling structure as well. (this structure is fixed).

1. Convolution block :

The main parameters to pass in this block are separable (if we need a mobilenet version then make this True). kernel\_quant, quantrelu for quantization. And img\_size to calculate the least number of layers we can take. (For now calculated number of layers based on image size).

1. Classification head (FC) :

The main parameters for this layer is kernel\_quant, dropout rate, metrics. This module also uses L2 regularizer as activity regularizer to make FC output values within 5.10 range (The value of L2 used is by experiments). The classification head uses Categorical cross-entropy for multi-class classification by default. Also for multi-class output Softmax activation is applied by default and for binary , sigmoid in AutoKeras. Hence the experiments are performed for multiclass classification (Because on hardware side we support only softmax activation as post-processing)

2. MODELS TRAINED SO FAR :

All the models trained using AutoKeras have resolution 32x32x1 and handgesture dataset is used for training (around 11,000 train images belonging to 11 classes and around 3000 test images). There are mainly two model variations, vanilla convolutional network and a mobilenet version of that. And another variation of that is models with kernel and activation quantization or without both quantization.

By performing these experiments, we received 87+ accuracy (in both mobilenet and non-mobilenet version) when using kernel and activation quantization. And without quantization we received 90+ accuracy. Each model variation was trained 3-4 to know what minimum accuracy we achieve. Some of the models are provided along with this document.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Mobilenet? | Quant? | accuracy | # params | device | MAE | Bin size  (KB) | Cycles consumption  (total) |
| No | Yes | 89% | 19,979 | Clnx  compact | 0.006 | 43 | 555,819 |
| Yes | Yes | 91% | 12,375 | Clnx  compact | 0.027 | 66 | 1,422,104 |
| Yes | No | 93% | 14,923 | ECP5 | 0.02 | 97 | 582,180 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

3. ENVIRONMENT TO TRAIN THE NETWORK:

The models were trained on Linux GPU using TF-GPU2.3 (requires cuda 10.1) and AK 1.0.15. Python 3.6.9 is used and the working environment was created using python environment using command “python3 –m venv path/to/new/virtual/env” and this python virtual env can be recreated using the attached requirement.txt file using “python3 –m pip install –r requirement.txt”

4. HOW TO TRAIN THE MODEL USING PROVIDED SCRIPT :

The table below shows which parameters in the script to change for which condition :

|  |  |  |
| --- | --- | --- |
| PARAMETER | PURPOSE |  |
| quantrelu | Activation quantization for conv |  |
| kernel\_quant | Kernel quantization for conv and FC |  |
| seperable | True if mobilenet version is to be trained |  |
| train\_and\_val\_separate | False if train and validation data is not separate |  |
|  |  |  |
|  |  |  |
|  |  |  |

Once the parameters are changed (if required) then run the provided script ak\_training.py (python ak\_training.py)

Once the model is done training make sure to check in the print logs that, dense/FC layer output is within -32 to +32 range to abide to 5.10 format

LIMITATIONS / RESTRICTIONS :

1. The model training was done considering only “CLASSIFICATION” task only.
2. The optimizer which Autokeras chooses sometimes has a very small initial learning rate and sometimes it is used along with learning rate decay and this affects training accuracy/loss. Hence a constant optimizer was used (SGD with initial LR=0.1 and with lr\_schedule)
3. For now the only hyperparameter which is varying is, the number of channels (depth) in each layer. If the “number of layers” is kept as a hyperparameter then it tries to go for a very large depth near FC layer and this creates FC output value exploding (they go beyond 5.10 limit which is our hardware limit for crosslink device). So the number of layers are for now fixated.
4. “Max model size” parameter is tested with few experiments (with given seed and resolution) to create a model (actually .bin file size) lesser than 250kB if using ultraplus device.
5. BATCH\_SIZE parameter is for creating train/validation data in the required format (with preprocessing) and the value used is greater than number of total images in training dataset. The actual batch size used during training is the default value used by AutoKeras which is 32.
6. For Reproducibility, when the seed is provided, it searches through same hyperparameter combinations everytime we run the script. However, the loss value which the AutoKeras get might differ slightly and as a result may not give the same architecture as earlier. But the accuracy remains approximately within +/- 3% range.