Automated CNN Architecture with Evolutionary Algorithm

Ruben Rodriguez Buchillon and Volha Leusha, {rubenr2, leusha}@stanford.edu

OVERVIEW

Objective of this project is to automate the process of finding Convolutional Neural Network (CNN) architecture with best possible accuracy and minimal training and prediction compute. Second objective is to research into possibilities to improve existing state of art Automated CNN Search based on evolutionary algorithms.

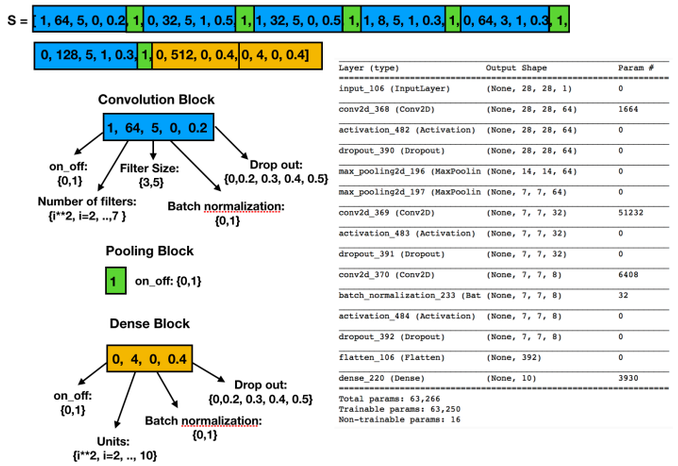
Background:
This project is based on Evolutionary Algorithm described in [1] and [2] that use test accuracy as fitness function and simple tournament strategy to sample children networks.

Problems: this approach potentially leads to local maximum, and is relatively slow

Expand existing research by:
- Using modified fitness function that takes into account not only accuracy, but also prediction time: potential reduction in run time (ideally without reduction in accuracy)
- Add Epsilon-Greedy and Gibbs sampling to tournament strategy: potentially solving local maxima problem and consequently improving accuracy

EVOLUTIONARY ALGORITHM

1. **Create n random sequences**
2. **Convert sequences to CNNs**



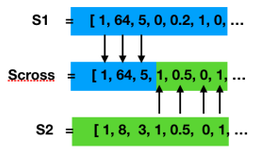
Layer (type)	Output Shape	Param #
input_106 (InputLayer)	(None, 28, 28, 1)	0
conv2d_368 (Conv2D)	(None, 28, 28, 64)	1664
activation_482 (Activation)	(None, 28, 28, 64)	0
dropout_390 (Dropout)	(None, 28, 28, 64)	0
max_pooling2d_196 (MaxPoolin	(None, 14, 14, 64)	0
max_pooling2d_197 (MaxPoolin	(None, 7, 7, 64)	0
conv2d_369 (Conv2D)	(None, 7, 7, 32)	51232
activation_483 (Activation)	(None, 7, 7, 32)	0
dropout_391 (Dropout)	(None, 7, 7, 32)	0
conv2d_370 (Conv2D)	(None, 7, 7, 8)	6408
batch_normalization_233 (Bat	(None, 7, 7, 8)	32
activation_484 (Activation)	(None, 7, 7, 8)	0
dropout_392 (Dropout)	(None, 7, 7, 8)	0
flatten_106 (Flatten)	(None, 392)	0
dense_220 (Dense)	(None, 10)	3930

Total params: 63,266
Trainable params: 63,250
Non-trainable params: 16

3. **For each CNN calculate fitness function**
   - fit1: {accuracy}
   - fit2: {accuracy, prediction time}
4. **Select next generation of n sequences using different sample strategies:** Random, Tournament, Gibbs, added Epsilon-Greedy, fit1, fit2, and combined
5. **Crossover, mutate**
6. **Go to step 2 and repeat for m iterations**

Crossover

1. Randomly choose index to cross two sequences : let idx = 3
2. Cross S1[:idx] and S2[idx:]



Mutation

For mutation in range(number of mutations):
Choose random point from S:
Mutate random point of S with probability p(i)

Number of mutations = 2:

S = [1, 64, 5, 1, 0.5, 0, 1, ...

Smut = [1, 32, 5, 0 0.5, 0, 1, ...

DATA

MNIST

- 28x28 grayscale
- 60000 train images
- 10000 test images
- 10 categories

Fashion-MNIST

Benchmark accuracy on the whole data set:

- Regularization of Neural Networks using Drop Connect: 0.21% error

Overused and too easy to train

- WRN-28-10+RE: 0.963
- Google AutoML with 24 hours search: 0.939
- VGG16 26M parameters: 0.935
- CNN 1.8M parameters: 0.932
- Budget models: 0.89

DISCUSSION

- Fully automated CNN architecture search model is created
- The model size is restricted to 6 convolutional and 2 dense layers due to computation constraints, however is extendable to more layers
- Gibbs and epsilon-greedy sampling added to tournament selection proved to prevent the model being stuck in the local maxima
- Modified fitness function proved to reduce run time and produce much smaller models with small drop in accuracy by ~ 1%

Best performance so far achieved with tournament epsilon: 93.7% accuracy with only 2h search and 1.7M parameters: better that VGG16 and CNN with 1.8M, and faster than Google AutoML

REFERENCES

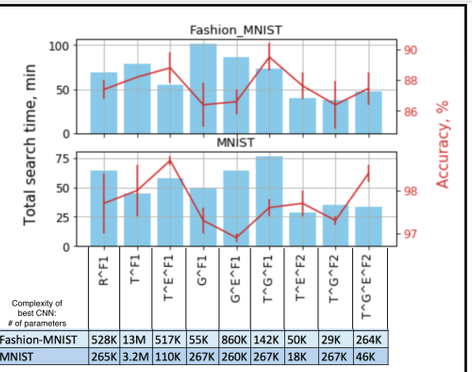
[1] Yanan Sun., Bing Xie, Mengjie Zhang, Gary G. Yen. Automatically Designing CNN Architectures Using Genetic Algorithm for Image Classification, 2019
[2] Petra Vidnerova, Roman Neruda. Evolution Strategies for Deep Neural Network Models Design, 2017
[3] Sebastian Heinz. A performance benchmark of Google AutoML Vision using Fashion-MNIST, 2018

EXPERIMENTS

Stage 1:

- 4 experiments for each sample strategy
- Both MNIST and Fashion- MNIST
- Uniform data sample: 5% of data
- Population size n =10, iterations m =10
- Only CPU: 3.1 GHz, 16GB RAM

Conclusions:
- Fit2 reduces training time by factor of two and produces smaller networks without significant drop in accuracy
- Epsilon-greedy in general helps model with tournament sampling to improve performance
- Gibbs sampling alone has low accuracy, but helps tournament strategy to improve accuracy
- Random sampling has the most unstable performance



Complexity of best CNN: # of parameters	R^F1	T^F1	T^E^F1	G^F1	G^E^F1	T^G^F1	T^E^F2	T^G^F2	T^G^E^F2
Fashion-MNIST	528K	13M	517K	55K	860K	142K	50K	29K	264K
MNIST	265K	3.2M	110K	267K	260K	267K	18K	267K	46K

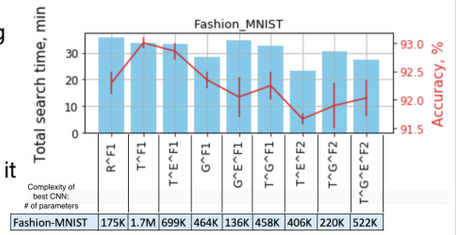
Sample strategies:

R	:Random	E	:Epsilon-greedy
T	:Tournament	F1	:fit1
G	:Gibbs	F2	:fit2

Stage 2:

- Use 100% of data and only Fashion-MNIST
- 2 experiments for each sample strategy: n=10, m=10
- GPU - Nvidia Tesla P100
- CPU acceleration: single vs multiprocess

Conclusions:
- Fit2 reduces training time, however the difference is less significant for GPU
- Fit2 produces smaller models, but it slightly drops best accuracy
- Best accuracy achieved by tournament strategy with/ without epsilon-greedy



Complexity of best CNN: # of parameters	R^F1	T^F1	T^E^F1	G^F1	G^E^F1	T^G^F1	T^E^F2	T^G^F2	T^G^E^F2
Fashion-MNIST	175K	1.7M	699K	464K	136K	458K	406K	220K	522K

Stage3:

- n=15, m=20, GPU
- Use 100% Fashion-MNIST
- Only T^E^F1 and T^E^G^F2

	Total search time, min	Best Accuracy	Complexity of best CNN: # of parameters
T^E^F1	117	93.7%	1.7M
T^G^E^F2	85	92.3%	242K

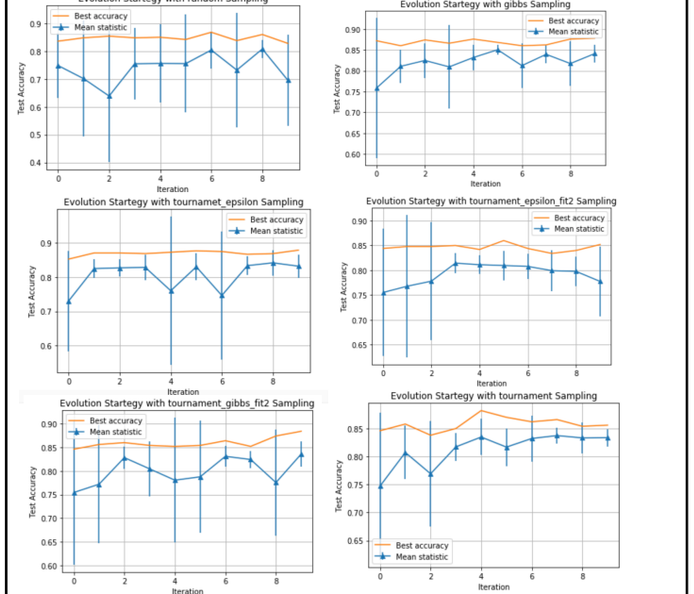
Conclusions:
- T^E achieved best accuracy of 93.7% with 1.7M parameters and 2h of search: better and smaller than VGG16, and faster than AutoML
- Fit2 function again proved to find smaller model with less search time, with drop in accuracy by ~1%

Stage4:

Next Steps: Increase epochs and iterations to outperform Google AutoML keeping search time under 3h

Stability Experiment:

Plot per iteration :
- Best accuracy trend
- Mean and standard deviation of all accuracies



- Random sampling is highly unstable: no guarantee to find a good model
- Tournament sampling has the most stable growing mean/best accuracy, however can get stuck in local maxima
- Adding epsilon-greedy helps to add instability to tournament and move it out from the local maxima
- Fit2 function improves train time, but can direct model away from the maximum: needs epsilon-greedy for correction
- Gibbs sampling alone produces worse results than the tournament, however can be used to stabilize other models