Multiobjective evolution for convolutional neural network architecture search

Petra Vidnerová  , Štěpán Procházka  ,

Roman Neruda 

Institute of Computer Science
The Czech Academy of Sciences

ICAISC Online Conference, October 2020

Outline

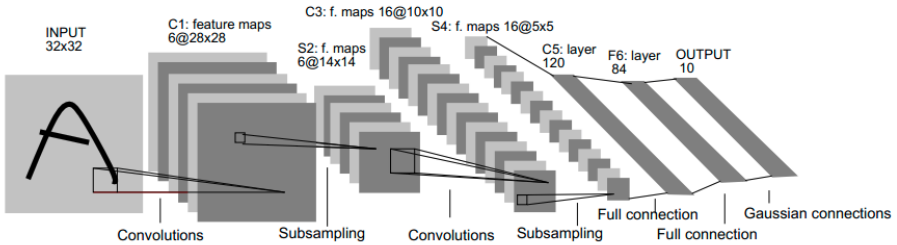
Evolving architectures

- Introduction
  - Deep Neural Networks
  - Neural Architecture Search
  - Related Work
- Our Approach
  - Evolutionary search
  - Multiobjective optimisation
- Experiments
  - MNIST, fashionMNIST

Introduction

Convolutional Neural Networks

- convolutional networks - networks with convolutional layers
- our work: feed-forward neural networks, fully connected

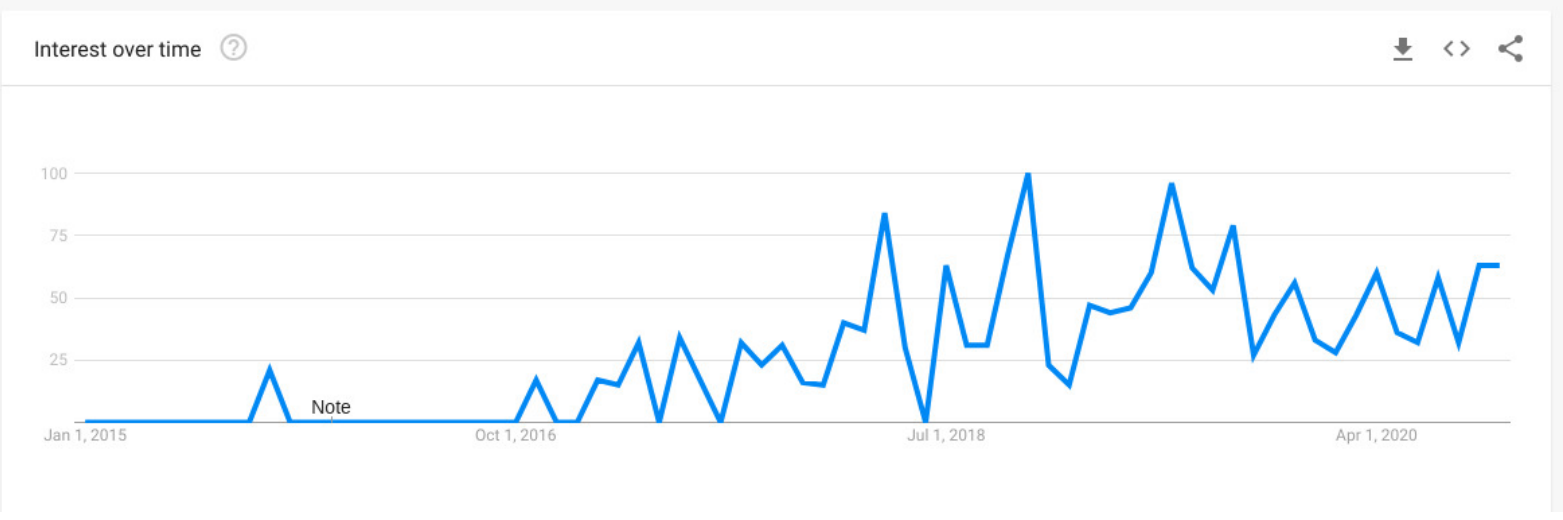


Network Architecture

- typically designed by humans
- trial and error method
- goal: automatic design

Neural architecture search

- established reseach field
- increasing interest recently due to more accesible computational resources



- so our reasearch is a drop in the ocean
- simplified task, goal is to verify the multiobjective approach

Related Work - evolutionary approaches

NEAT - NeuroEvolution of Augmenting Topologies

- Ken Stanley, 2002,
  `www.cs.ucf.edu/~kstanley/neat.html`

Works focused on parts of network design of DNNs

- I. Loshchilov and F. Hutter, 2016
  CMA-ES for hyperparameter optimization of deep neural networks

Optimising deep architectures through evolution

- DeepNEAT - extending NEAT do deep networks
  R. Miikkulainen, J. Z. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat, *Evolving deep neural networks,* 2017, CoDeepNEAT

Related Work - software tools

AutoKeras

- software library for automated machine learning, automated architecture and hyperparameter search, Baysesian optimisation

Talos

- semiautomatic approach to hyperparameter search for Keras models, based on grid search

hyperopt

- from 2013, a Python library that enables distributed hyper-optimisatin optimisation

Our Approach

- focus on feedforward deep neural networks and convolutional networks
- this paper is limited to convolutional networks only

- keep the search space as simple as possible
- only architecture is optimized, weights are learned by gradient based technique
- the approach is inpired by and designed for KERAS library
- architecture defined as list of layers, layer is fully connected layer (dense layer) or convolutional layer
- layer defined by number of neurons/filters, activation function, type of regularization

Evolutionary Algorithms

- a population of *individuals* representing feasible solutions
- each individual has assigned a *fitness* value
- population evolves by means of *selection, crossover, and mutation*

Evolving convolutional networks

- individuals consists of two parts: convolutional and dense
- convolutional block - number of filters, kernel size, activation function type
- pool-size
- dense block - number of neurons, activation function type, regularization
- crossover convolutional and dense part separately
- mutation alters values, delete and adds blocks

Towards multiobjective optimisation

Fitness

- evaluate crossvalidation error on trainset
- fit network using gradient based technique

Problems ecountered

- sometimes architecture bloats even if it is not needed
- large architectures require larger computational time and slow down the search process
- smaller solutions are more valueable

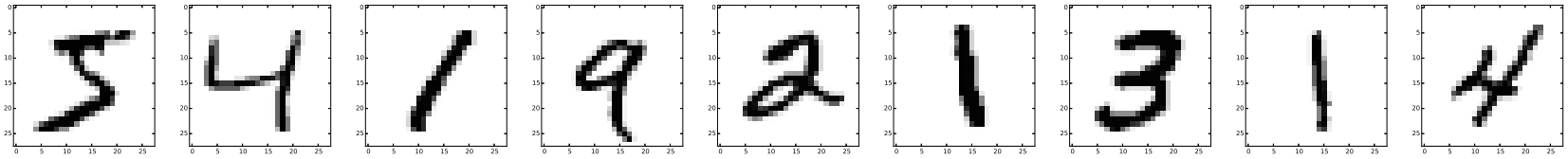
Solution

- multiobjective evolution - minimize crossvalidation error, minimize network size
- NSGAII

Experiments

Data Set

- well known data set MNIST, classification of hand written digits
- 28×28 pixels, 60000 for training, 10000 for testing



- fashionMNIST, classification of fashion objects

Results

Classification accuracy

Task	baseline	GA-CNN		NSGA-CNN	
		avg	std	avg	std
MNIST	98.97	99.19	0.26	99.36	0.02
Fashion-MNIST	91.64	93.13	0.20	92.67	0.42

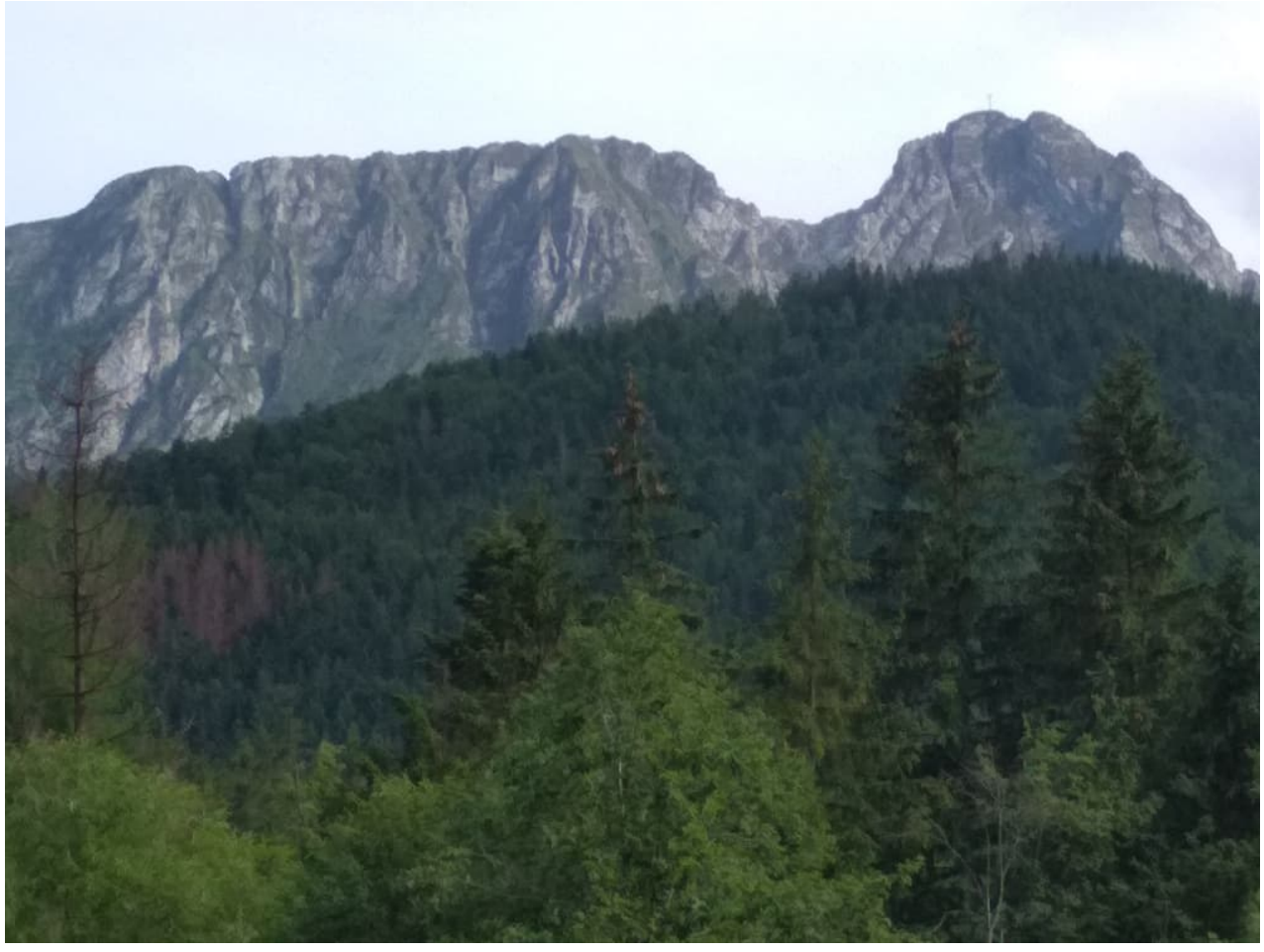
Network parameters

Task	baseline	GA-CNN		NSGA-CNN	
		avg	std	avg	std
MNIST	600K	690K	399K	77K	48K
Fashion-MNIST	356K	769K	339K	418K	311K

Conclusion

- an approach for neural architecture search for convolutional networks proposed
- the performance of the network and network size optimised simultaneously
- produces competitive networks of reasonable size
- prevents the candidate solutions to bloat and slow down the evolution
- the smaller solutions learn faster and are suitable for devices with limited memory
- Python implementation, works both on CPU and GPU (able to train several networks on one GPU)
  `https://github.com/PetraVidnerova/nsga-keras`

Thank you. Questions?



Hope to see you in Zakopane in 2021!