Master in Computer Vision Barcelona

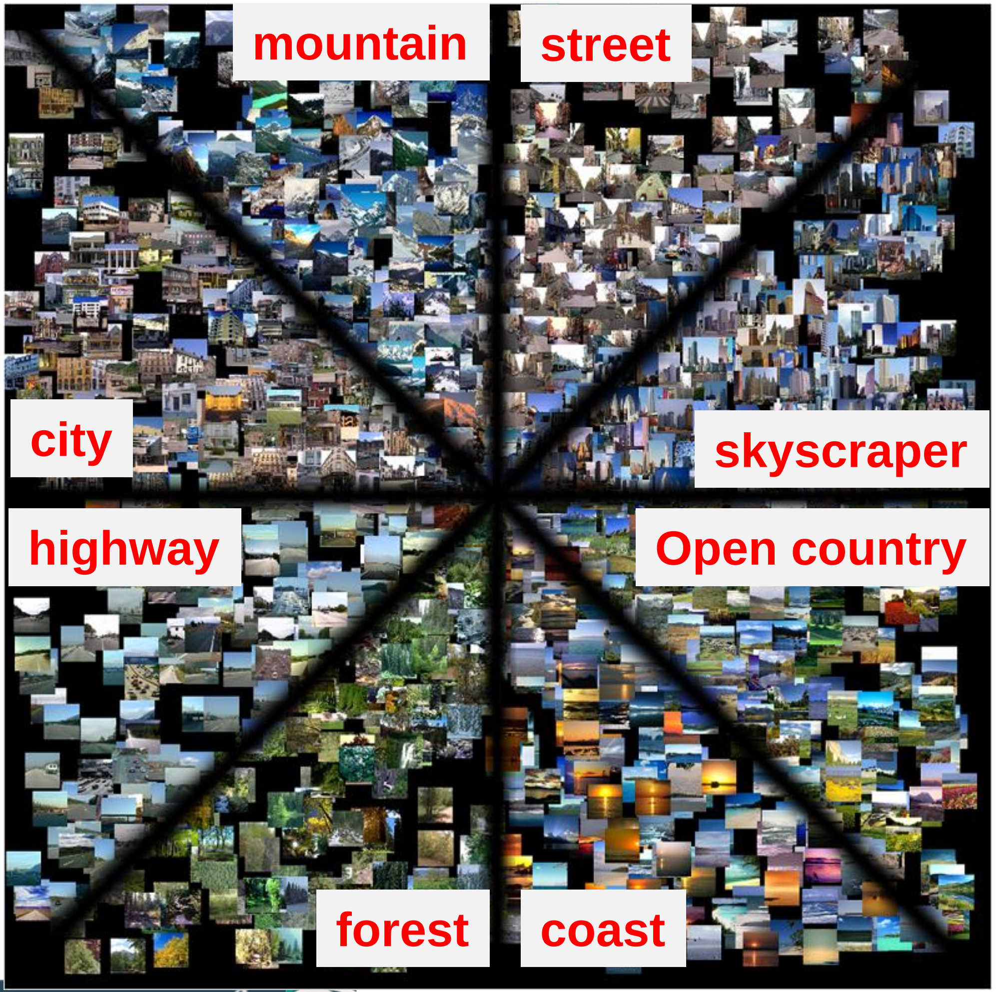
Module 3: Machine Learning for Computer Vision

Project: Deep learning classification

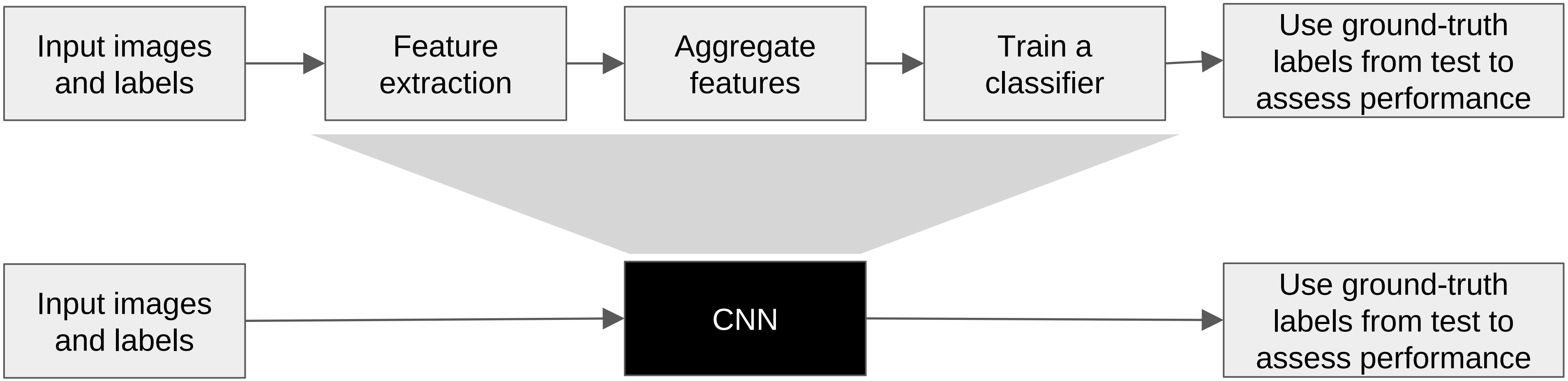
Lecturer: Ramon Baldrich

Module Goal

The aim of this module is to learn the techniques for category classification: handcrafted and learned.



Pipeline of the project W5 and W6



End to end task

Machine learning for image classification:
- Data driven methods: Deep Convolutional Networks: 3 sessions
  - From hand-crafted to learnt features
  - Fine tuning of pre-trained CNNs
  - Training a CNN from scratch

Keras: first example

```
# create model
model = Sequential()
model.add(Dense(12, input_dim=8, init='uniform', activation='relu'))
model.add(Dense(8, init='uniform', activation='relu'))

inputs = Input(shape=None))
x = Dense(12, init='uniform', activation='relu', name='fc1')(x)
x = Dense(8, init='uniform', activation=‘sigmoid', name= 'predictions’)(x)
model = Model(inputs, x, name=‘example')
```

W3-5

```
# Compile model
model.compile(loss=‘categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Fit the model
model.fit(X, Y, nb_epoch=150, batch_size=10)
# evaluate the model
scores = model.evaluate(X, Y)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

W3-4

```
# predict with the model
features = model.predict(X)
```

W3-4

Week 5: Training a CNN from scratch

1

Return of the Devil in the Details: Delving Deep into Convolutional Nets

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman
Visual Geometry Group, Department of Engineering Science, University of Oxford
{ken,karen,vedaldi,az}@robots.ox.ac.uk

Abstract—The latest generation of Convolutional Neural Networks (CNN) have achieved impressive results in challenging benchmarks on image recognition and object detection, significantly raising the interest of the community in these methods. Nevertheless, it is still unclear how different CNN methods compare with each other and with previous state-of-the-art shallow representations such as the Bag-of-Visual-Words and the Improved Fisher Vector. This paper conducts a rigorous evaluation of these new techniques, exploring different deep architectures and comparing them on a common ground, identifying and disclosing important implementation details. We identify several useful properties of CNN-based representations, including the fact that the dimensionality of the CNN output layer can be reduced significantly without having an adverse effect on performance. We also identify aspects of deep and shallow methods that can be successfully shared. In particular, we show that the data augmentation techniques commonly applied to CNN-based methods can also be applied to shallow methods, and result in an analogous performance boost. Source code and models to reproduce the experiments in the paper is made publicly available.

arXiv:1405.3531v4 [cs.CV] 5 Nov 2014

1 Introduction

Perhaps the single most important design choice in current state-of-the-art image classification and object recognition systems is the choice of visual features, or image representation. In fact, most of the quantitative improvements to image

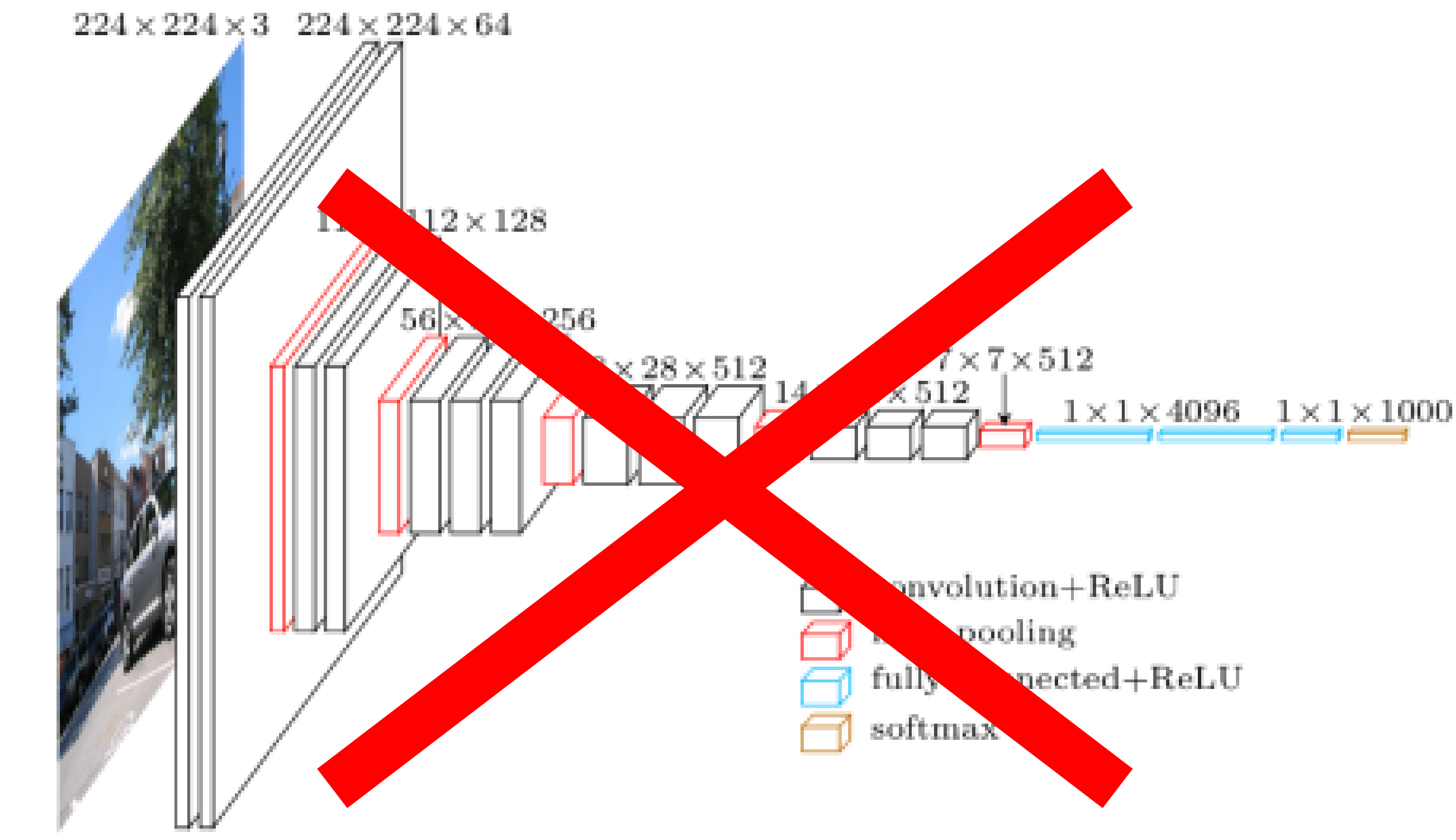
is handcrafted, they contain a very large number of parameters learnt from data. When applied to standard image classification and object detection benchmark datasets such as ImageNet ILSVRC [5] and PASCAL VOC [6] such networks have demonstrated excellent performance [7], [8], [9], [10], [11], significantly better than standard image encod-

Goals:

- Putting all together
- Pipeline Control
- Taking decisions

Chatfield, Ken, et al. "Return of the devil in the details: Delving deep into convolutional nets." *arXiv preprint arXiv:1405.3531* (2014).

Very deep convolutional networks for large-scale image recognition



Credit Davi Frossard

layers

```
x = Convolution2D(…)(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='pool2')(x)

x = Flatten(name='flatten')(x)
x = Dense(…, activation='relu’)(x)
x = Dense(…, activation='softmax')(x)
```

…. more layers

```
x = BatchNormalization()(x)
x = LayerNormalization()(x)
x = Dropout()(x)
```

…. even more layers

```
x = GaussianNoise (..)(x)
x = Activation()(x)
```

Layers API overview

Preprocessing layers
- Image preprocessing layers
- Image augmentation layers

Normalization layers
- BatchNormalization layer
- LayerNormalization layer
- UnitNormalization layer
- GroupNormalization layer

Regularization layers
- Dropout layer
- SpatialDropout2D layer
- GaussianDropout layer
- GaussianNoise layer
- ActivityRegularization layer
- AlphaDropout layer

Layer activations
- relu function
- sigmoid function
- softmax function
- softplus function
- softsign function
- tanh function
- selu function
- elu function
- exponential function

Layer weight initializers
- RandomNormal class
- RandomUniform class
- TruncatedNormal class
- Zeros class
- Ones class
- GlorotNormal class
- GlorotUniform class
- HeNormal class
- HeUniform class
- Identity class
- Orthogonal class
- Constant class
- VarianceScaling class

Layer weight regularizers
- L1 class
- L2 class
- L1L2 class
- OrthogonalRegularizer class

Layer weight constraints
- MaxNorm class
- MinMaxNorm class
- NonNeg class
- UnitNorm class
- RadialConstraint class

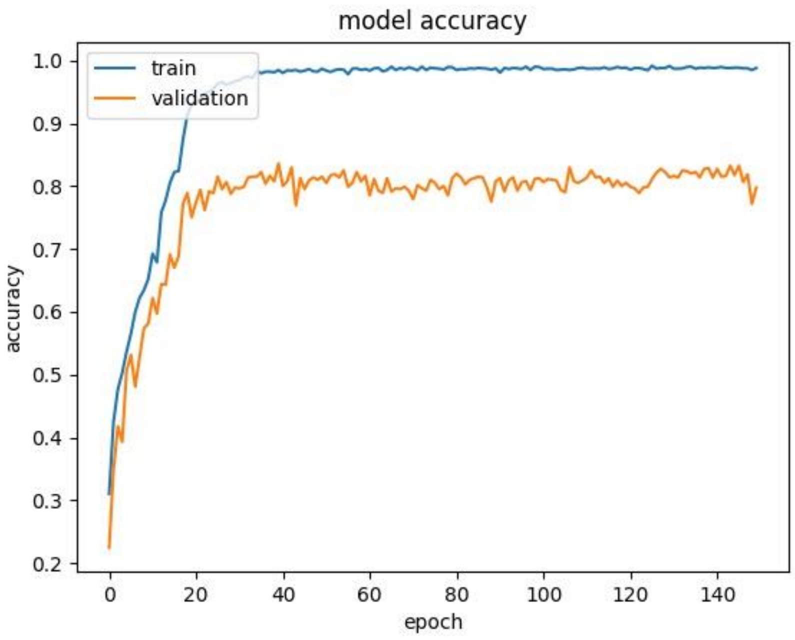
Core layers
- Input object
- Dense layer
- Activation layer
- Embedding layer
- Masking layer
- Lambda layer

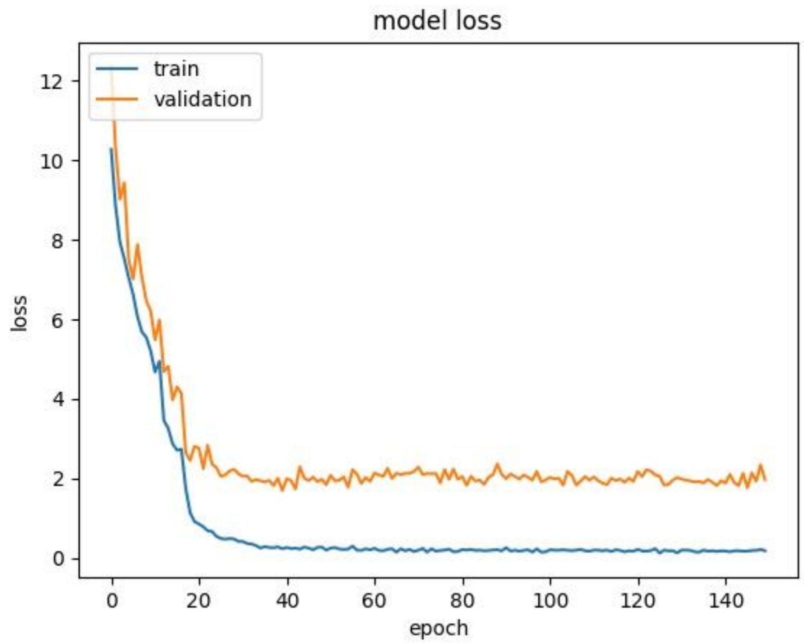
Convolution layers
- Conv1D layer
- Conv2D layer
- SeparableConv1D layer
- SeparableConv2D layer
- DepthwiseConv2D layer
- Conv1DTranspose layer
- Conv2DTranspose layer

Pooling layers
- MaxPooling1D layer
- MaxPooling2D layer
- MaxPooling3D layer
- AveragePooling1D layer
- AveragePooling2D layer
- GlobalMaxPooling1D layer
- GlobalMaxPooling2D layer
- GlobalAveragePooling1D layer
- GlobalAveragePooling2D layer

Baseline

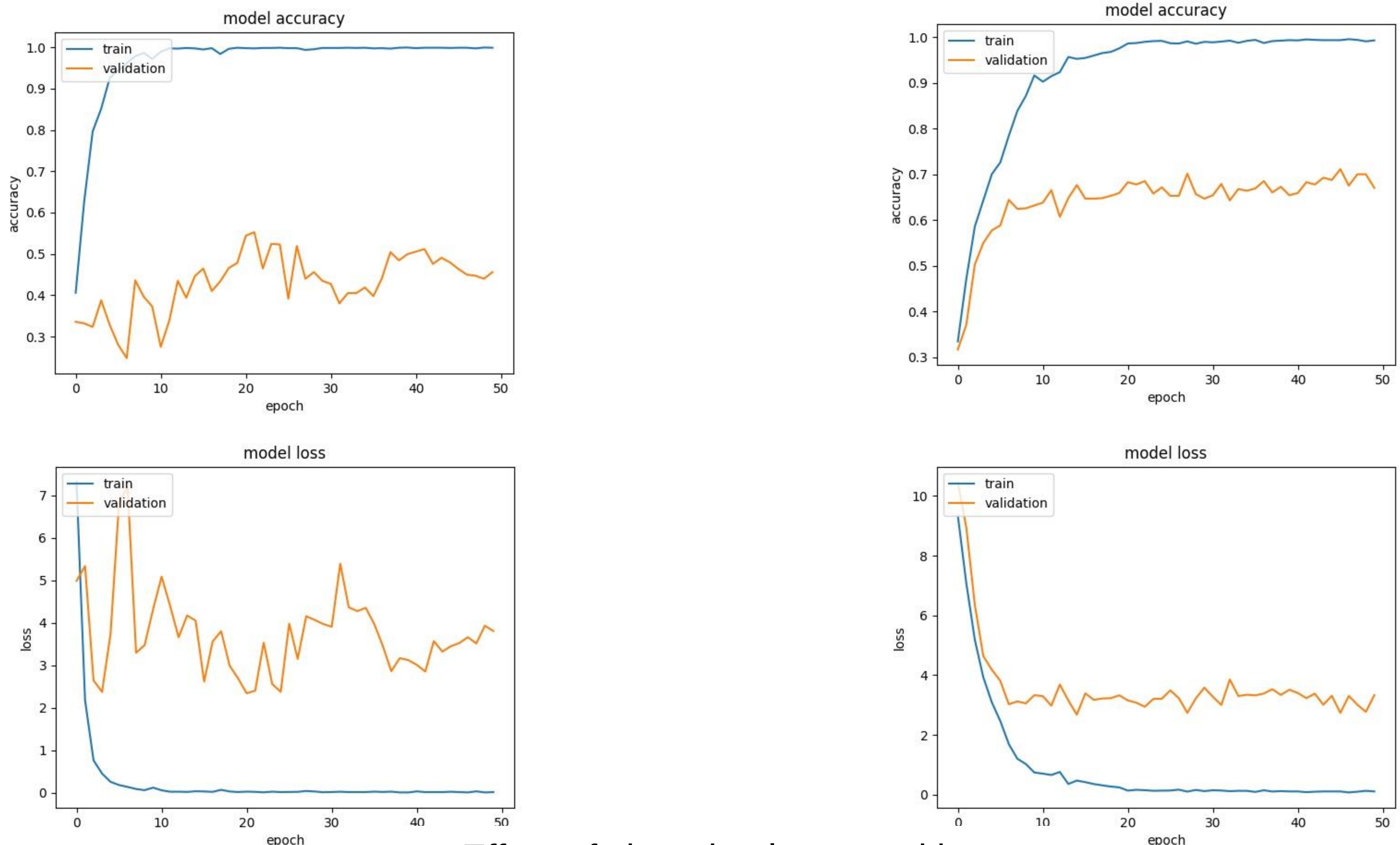
2 convolutional layers + …





examples

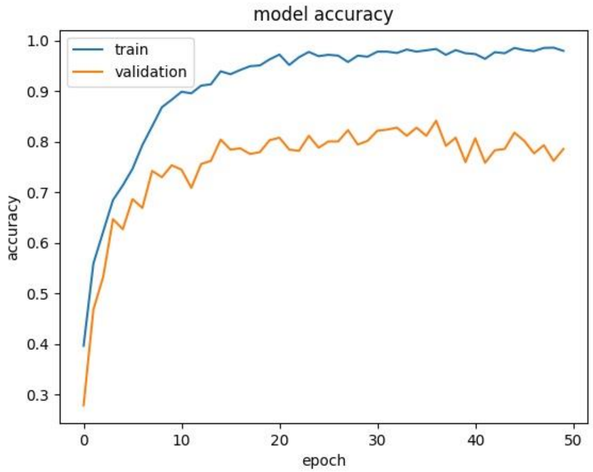
2 convolutional layers + batch normalization + …

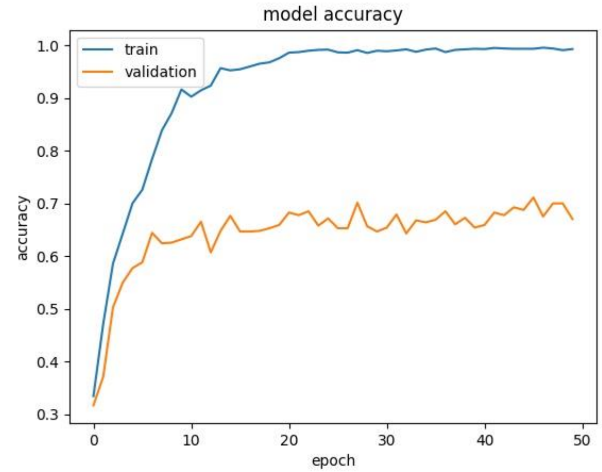


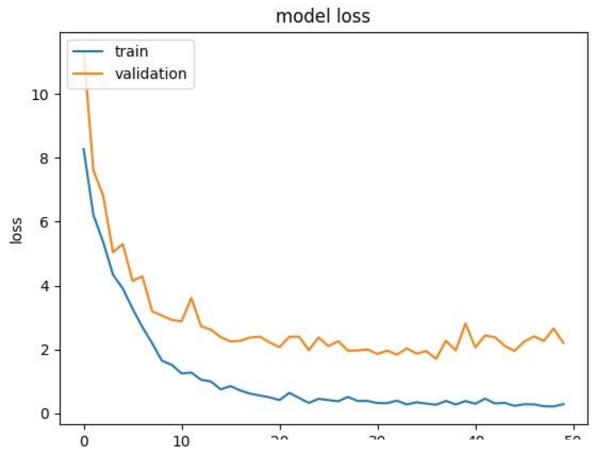
Effect of changing layer position

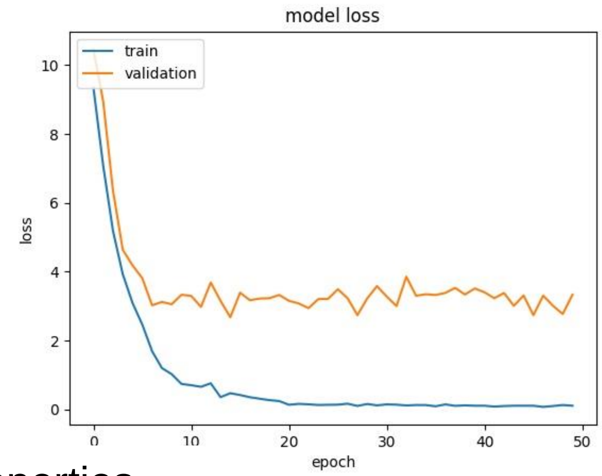
examples

2 convolutional layers + dropout + …









Effect of changing layer position or properties

Control model size

```
model.summary()
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 128, 128, 3)   0
____________________________________________________________________________________________________
                                 (None, 128, 128, 32)  2432        input_1[0][0]
                                 ____________________________________________________________________
                                 (None, 64, 64, 32)    0
                                 ________________________________
                                 (None, 64, 64, 32)    128
                                 ________________________________
                                 (None, 64, 64, 64)    51264
                                 ________________________________
                                 (None, 32, 32, 64)    0
                                 ________________________________
                                 (None, 32, 32, 64)    256
                                 ________________________________
                                 (None, 16, 16, 64)    0
                                 ________________________________
                                 (None, 16384)         0
                                 ________________________________
                                 (None, 4096)          67112960
_________________________________________________________________
predictions (Dense)              (None, 8)             32776
====================================================================================================
Total params: 67,199,816
Trainable params: 67,199,624
Non-trainable params: 192
____________________________________________________________________________________________________
```

Modern classification architectures

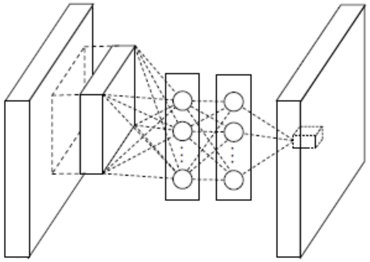
Network In Network

Min Lin[1,2], Qiang Chen[2], Shuicheng Yan[2]
[1]Graduate School for Integrative Sciences and Engineering
[2]Department of Electronic & Computer Engineering
National University of Singapore, Singapore
{linmin,chenqiang,eleyans}@nus.edu.sg

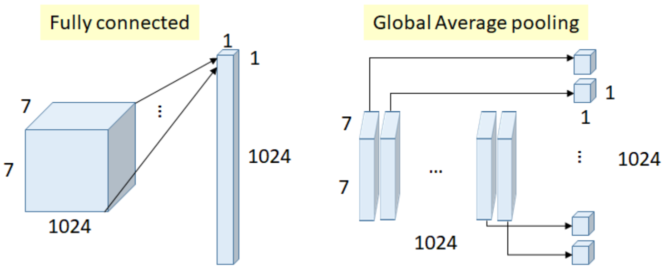
Abstract

We propose a novel deep network structure called "Network In Network"(NIN) to enhance model discriminability for local patches within the receptive field. The conventional convolutional layer uses linear filters followed by a nonlinear activation function to scan the input. Instead, we build micro neural networks with more complex structures to abstract the data within the receptive field. We instantiate the micro neural network with a multilayer perceptron, which is a potent function approximator. The feature maps are obtained by sliding the micro networks over the input in a similar manner as CNN; they are then fed into the next layer. Deep NIN can be implemented by stacking mutiple of the above described structure. With enhanced local modeling via the micro network, we are able to utilize global average pooling over feature maps in the classification layer, which is easier to interpret and less prone to overfitting than traditional fully connected layers. We demonstrated the state-of-the-art classification performances with NIN on CIFAR-10 and CIFAR-100, and reasonable performances on SVHN and MNIST datasets.

- MLPConv

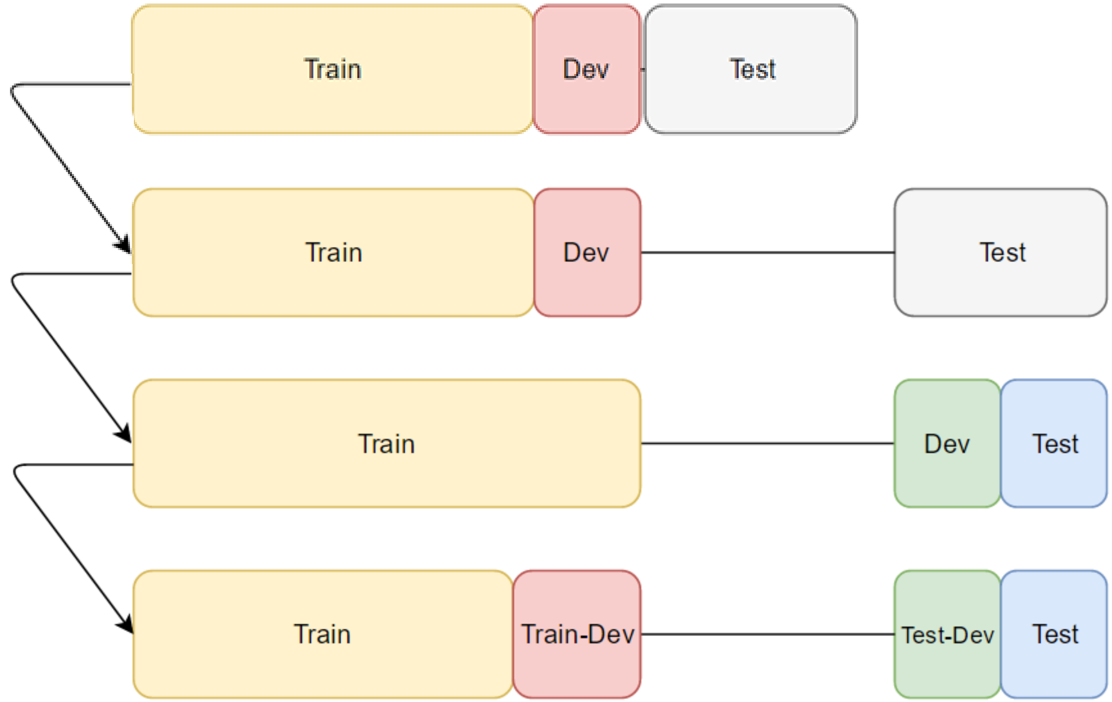


- Global Average Pooling Layer



https://www.coursera.org/lecture/convolutional-neural-networks/networks-in-networks-and-1x1-convolutions-ZTb8x

Bias-Variance



Nuts and Bolts of Applying Deep Learning (Andrew Ng)

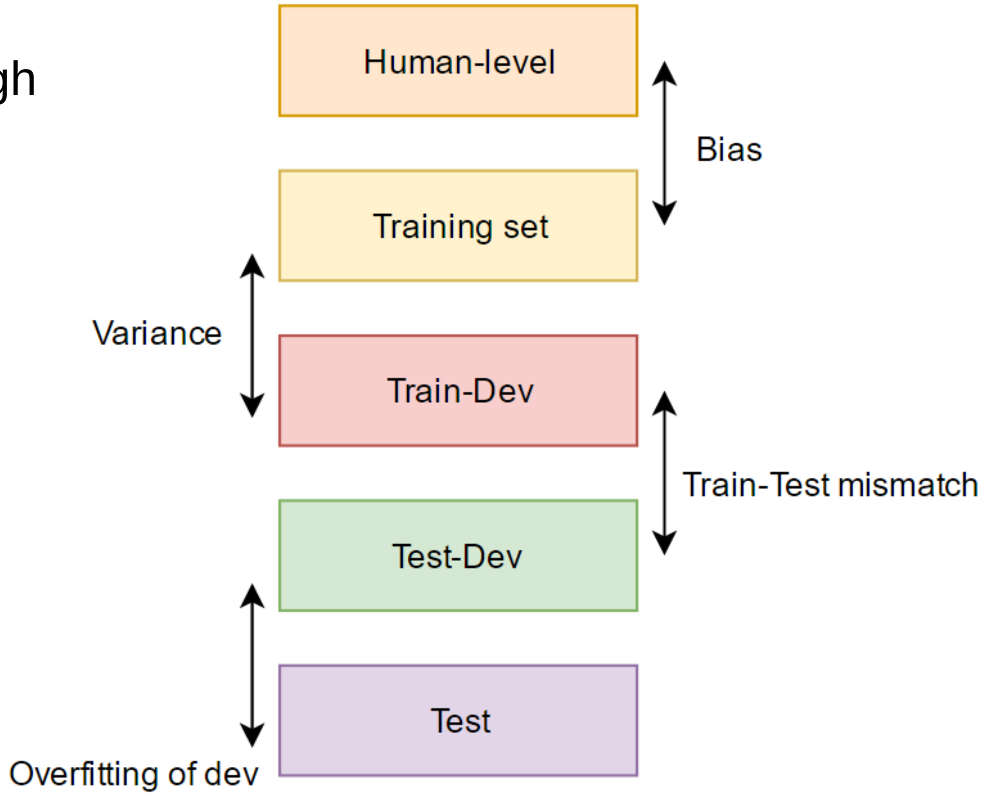
https://www.youtube.com/watch?v=F1ka6a13S9I&t=3044s

https://kevinzakka.github.io/2016/09/26/applying-deep-learning/

Bias-Variance

The **bias** is error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

The **variance** is error from sensitivity to small fluctuations in the training set. High variance can cause overfitting: modeling the random noise in the training data, rather than the intended outputs.

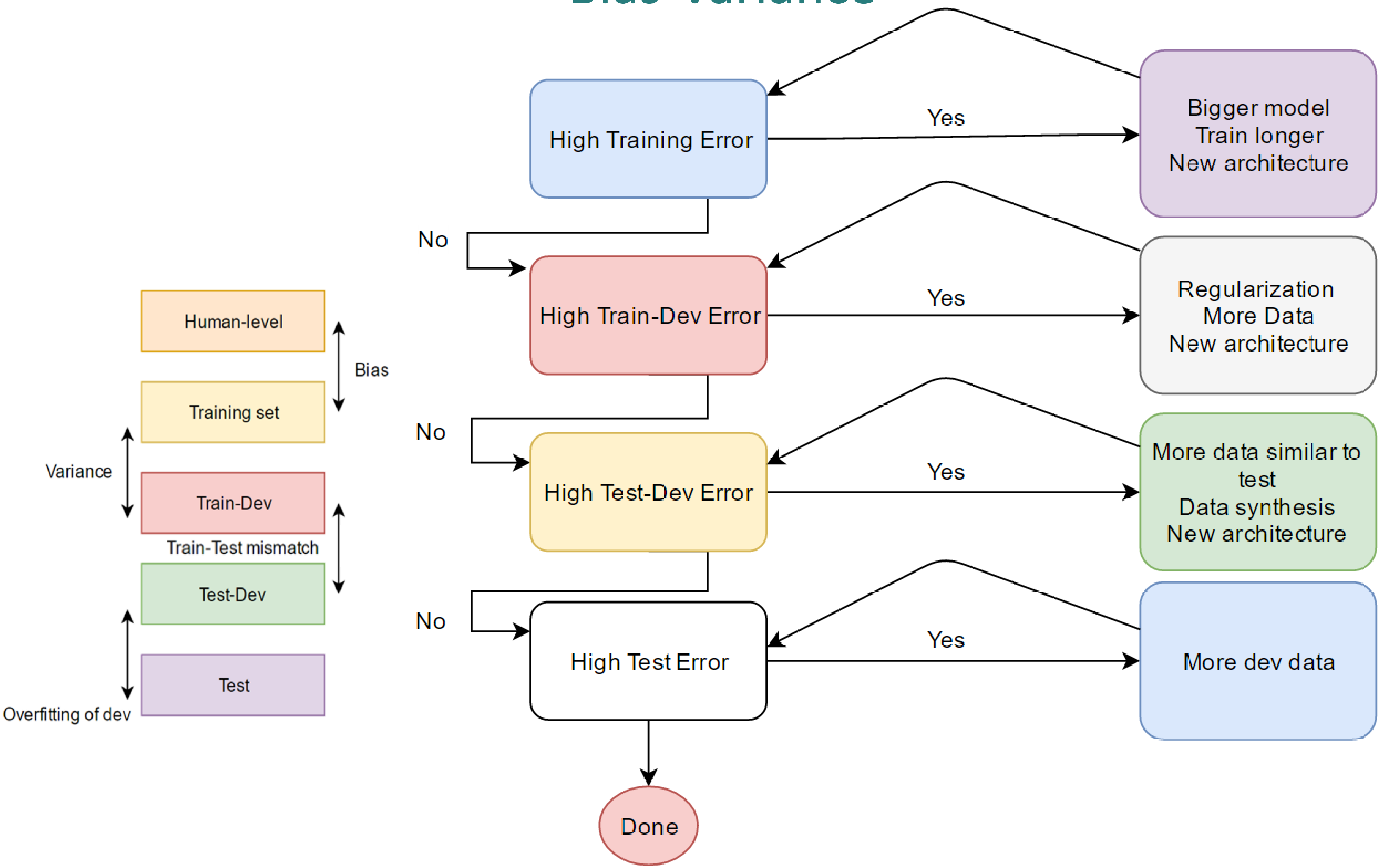


Nuts and Bolts of Applying Deep Learning (Andrew Ng)

https://www.youtube.com/watch?v=F1ka6a13S9I&t=3044s

https://kevinzakka.github.io/2016/09/26/applying-deep-learning/

Bias-Variance



Make it work

What we will look at?

- compact model (few parameters)
- few convolutional layers
- different layer types involved
- performance
- ratio: accuracy/ (number of parameters/ 100K)

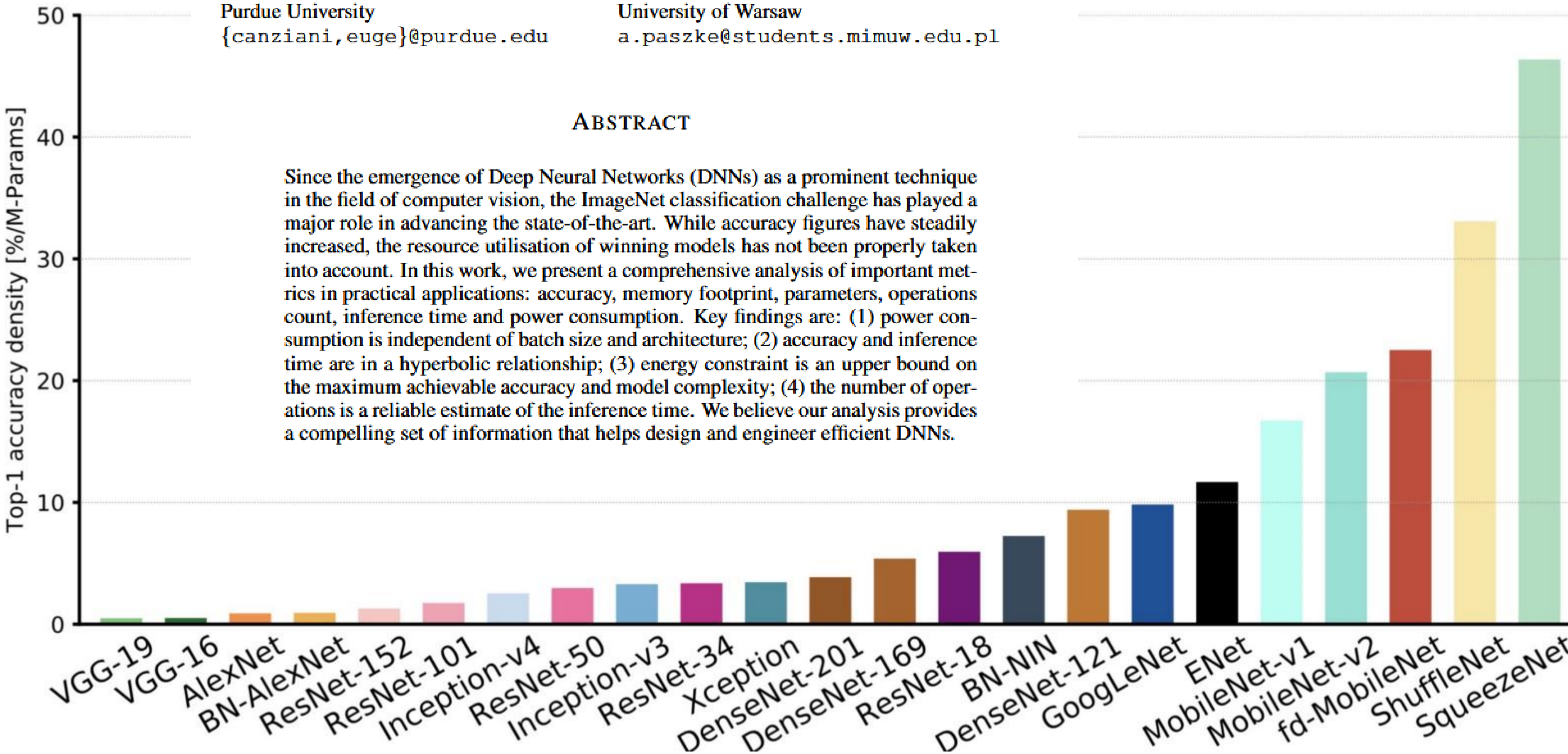
An Analysis of Deep Neural Network Models for Practical Applications

Alfredo Canziani & Eugenio Culurciello
Weldon School of Biomedical Engineering
Purdue University
{canziani,euge}@purdue.edu

Adam Paszke
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw
a.paszke@students.mimuw.edu.pl

Abstract

Since the emergence of Deep Neural Networks (DNNs) as a prominent technique in the field of computer vision, the ImageNet classification challenge has played a major role in advancing the state-of-the-art. While accuracy figures have steadily increased, the resource utilisation of winning models has not been properly taken into account. In this work, we present a comprehensive analysis of important metrics in practical applications: accuracy, memory footprint, parameters, operations count, inference time and power consumption. Key findings are: (1) power consumption is independent of batch size and architecture; (2) accuracy and inference time are in a hyperbolic relationship; (3) energy constraint is an upper bound on the maximum achievable accuracy and model complexity; (4) the number of operations is a reliable estimate of the inference time. We believe our analysis provides a compelling set of information that helps design and engineer efficient DNNs.



https://medium.com/@culurciello/analysis-of-deep-neural-networks-dcf398e71aae

deliverables and deadlines

Deliverable: presentation slides Monday 13th at 15:00

Final presentation:

groups 1,2,3,4,5: Monday 13th at 16:00

groups 6,7,8,9,10: Monday 13th at 17:30

12 + 3 minutes. Summarizing the whole work.

be specific, enhance your approach

Your questions will be graded

FINAL REPORT deadline: Saturday 18th at 23:00