LEARNING DEEP MULTI-MODAL FUSION ARCHITECTURES

GRAHAM TAYLOR
SCHOOL OF ENGINEERING
UNIVERSITY OF GUELPH

Joint work with:
Natalia Neverova (Facebook AI Research), Christian Wolf (INSA-Lyon)
Dhanesh Ramachandram (Guelph), Michal Lisicki (Guelph)
Timothy Shields (SRI), Mohamed Amer (SRI)

CVPR 2016 - WORKSHOP ON DEEP LEARNING FOR VISION
Las Vegas, NV
Seeing humans

- Humans: dominant subject in nearly all video
- Better algorithms for interpreting their behaviour can
  - help understanding of people’s use of public spaces
  - improve healthcare delivery and outcomes
  - augment people’s interaction with the world
  - improve human-computer and human-robot interaction
Recognizing intentional gestures

Dataset: Montalbano Gesture Recognition Dataset
Created for Chalearn 2014 “Looking at People” Competition

PhD work of Natalia Neverova and co-advisor Christian Wolf (while at INSA-Lyon)
Montalbano dataset details

- 13,858 instances of Italian conversational gestures performed by different subjects (20 categories)
- Recorded using consumer RGB-D sensor
- Color, depth video, articulated pose streams
- Audio stream added post-competition
Challenge #1:

Learn representations at multiple spatial and temporal scales.

This gesture can be fully characterized by upper-body motion.
Challenge #1:

Learn representations at multiple spatial and temporal scales.

This gesture can be fully characterized by upper-body motion.

Here, subtle finger movements play the primary role.
Challenge #2:

Integrate modalities.
Challenge #2:

Integrate modalities.
Challenge #3:

Training a complex model when data is not at web-scale.
Challenge #3:

Training a complex model when data is not at web-scale.
A multi-scale architecture

Fig. 2. The deep convolutional multi-modal architecture operating at 3 temporal scales corresponding to dynamic poses of 3 different durations. Although the audio modality is not present in the 2014 ChaLearn Looking at People Challenge dataset, we have conducted additional experiments by augmenting the visual signal with audio recordings from the 2013 version of the data.

In [43] the authors propose a late fusion strategy compensating for errors of individual classifiers by minimising the rank of a score matrix, and in a follow up work [44] identify sample-specific optimal fusion weights by enforcing similarity in fusion scores for visually similar labeled and unlabelled samples. Xu et al. introduced the Feature Weighting via Optimal Thresholding (FWOT) algorithm [45] jointly optimising feature weights and thresholds. In [46] MKL-based combinations of features act together with Bayesian model combination and weighted average fusion of scores from multiple systems.

A number of deep architectures have recently been proposed specifically for multi-modal data. Ngiam et al. [47] employ sparse RBMs and bimodal deep autoencoders for learning cross-modality correlations in the context of audio-visual speech classification of isolated letters and digits. Srivastava et al. [48] use a multi-modal deep Boltzmann machine in a generative fashion to tackle the problem of integrating image data and text annotations. Kahou et al. [7] won the 2013 Emotion Recognition in the Wild Challenge by building two convolutional architectures on several modalities, such as facial expressions from video frames, audio signal, scene context and features extracted around mouth regions. Finally, in [49] the authors propose a multi-modal convolutional network for gesture detection and classification from a combination of depth, skeletons and audio.

On a dataset such as ChaLearn 2014, we face several key challenges: learning representations at multiple spatial and temporal scales, integrating the various modalities, and training a complex model when the number of labeled examples is not at web-scale like static image datasets (e.g. [3]). We start by describing how the first two challenges are overcome at an architectural level. Our training strategy to overcome the last challenge is described in Sec. 4.

Our proposed multi-scale deep neural model consists of a combination of single-scale paths connected in a parallel way (see Fig. 2). Each path independently learns a representation and performs gesture classification at its own temporal scale given input from RGB-D video and articulated pose descriptors (audio channel can be also added, if available). Predictions from all paths are then aggregated through additive late fusion. This strategy allows us to first extract the most salient (in a discriminative sense) motions at a fine temporal resolution and, at the same time, consider them in the context of global gesture structure, smoothing and compensating for per-block errors typical for a given gesture class.

To differentiate among temporal scales, a notion of dynamic pose is introduced. By dynamic pose we mean a sequence of video frames, synchronized across modalities, sampled at a given temporal step \( s \) and concatenated to form a spatio-temporal 3d volume. Varying the value of \( s \) allows the model to leverage multiple temporal scales for prediction, thereby accommodating differences in tempos and styles of articulation of different users. Our model is therefore different from the one proposed in [4], where by...
**Single-scale deep architecture**

- **Path V1:**
  - Depth video, right hand
  - Intensity video, right hand

- **Path V2:**
  - Depth video, left hand
  - Intensity video, left hand

- **Path M:**
  - Mocap stream

- **Path A:**
  - Audio stream

- **Shared hidden layer (HLS)**
  - Max pooling
  - ConvD1
  - ConvC1
  - ConvD2
  - HLV1
  - HLV2
  - HLV3
  - ConvA1
  - ConvC1
  - ConvC2
  - HLM1
  - HLM2
  - HLM3
  - HLA1
  - HLA2

**DeepVision**

Learning Multimodal Fusion / G Taylor
Depth Video Stream

- Interested in capturing fine movements of palms and fingers
- Extract a bounding box around RHand, LHand centred at hand positions provided by skeleton
- Subtract background by thresholding along depth axis
- Apply local contrast normalization
Articulated Pose: Input

- Extract 11 joints from full-body skeleton (Kinect)
- **Position normalization**: HipCentre is an origin of a body-centred co-ordinate system
- **Size normalization** by the mean distance between each pair of joints (compensate for different body sizes, proportions, and shapes)
- Final representation (183-D descriptor)
  - Joint positions, velocities, and accelerations
  - Inclination angles
  - Azimuth angles
  - Bending angles
  - Pairwise distances

Training algorithm

- **Difficulties:**
  - Number of parameters:
    - ~12.4M per scale
    - ~37.2M total
  - Number of training gestures: ~10,000
- **Proposed solution:**
  - Structured weight matrices
  - Pretraining of individual channels separately
  - Careful initialization of shared layers
  - Iterative training algorithm which gradually increases # of parameters
Initialization: structured weights

- Top hidden layer from each path is initially wired to a subset of neurons in the shared layer.
- During fusion, additional connections between paths and the shared hidden layer are added.
Slightly different view

Blocks of the weight matrices are learned iteratively after proper initialization of the diagonal elements.
2014 ChaLearn Looking at People Challenge (ECCV)

Metric is mean Jaccard Index (intersection over union)
Post-competition improvements
Gesture localization

An additional binary classifier is employed for filtering and refinement of temporal position of each gesture.

The boundaries for each spotted gesture are extended or shrunk towards the closest switching point produced by the binary classifier.
Dropout (review)

- Introduced in 2012, made famous by ImageNet, now ubiquitous
- During training, for each training sample, “drop out” ~50% of hidden unit activities
- Punishes co-adaptation of units
- Can be viewed as very efficient model averaging
Moddrop - dropout on shared layer

\[ h_j^{(k)} = \sigma \left[ \sum_{i=1}^{F_k} w_{i,j}^{(k,k)} x_i^{(k)} + \gamma \sum_{n=1}^{K} \sum_{i=1}^{F_n} w_{i,j}^{(n,k)} x_i^{(n)} + b_j^{(k)} \right] \]
Moddrop: modality-wise dropout

- Punish co-adaptation of individual units (like dropout)
- Train a network which is robust/resistant to dropping of individual modalities (e.g. audio failure)

\[
\bar{h}^{(k)}_j = \sigma \left[ \sum_{i=1}^{F_k} w^{(k,k)}_{i,j} x_i^{(k)} + \sum_{\substack{n=1 \atop n \neq k}}^{K} \delta^{(k)} \sum_{i=1}^{F_n} w^{(n,k)}_{i,j} x_i^{(n)} + b^{(k)}_j \right]
\]

Bernoulli selector

\[
P(\delta^{(k)} = 1) = p^{(k)}
\]
## Moddrop results

**Classification accuracy on the validation set**
*(dynamic poses)*

<table>
<thead>
<tr>
<th>Modalities</th>
<th>Dropout (%)</th>
<th>Dropout + Moddrop (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>96.77</td>
<td>96.81</td>
</tr>
<tr>
<td>Mocap missing</td>
<td>38.41</td>
<td>92.82</td>
</tr>
<tr>
<td>Audio missing</td>
<td>84.10</td>
<td>92.59</td>
</tr>
<tr>
<td>Hands missing</td>
<td>53.13</td>
<td>73.28</td>
</tr>
</tbody>
</table>

**Jacquard index on test set (full gestures)**

<table>
<thead>
<tr>
<th>Modalities</th>
<th>Dropout (%)</th>
<th>Dropout + Moddrop (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>87.6</td>
<td><strong>88.1</strong></td>
</tr>
<tr>
<td>Mocap missing</td>
<td>30.6</td>
<td>85.9</td>
</tr>
<tr>
<td>Audio missing</td>
<td>78.9</td>
<td>85.4</td>
</tr>
<tr>
<td>Hands missing</td>
<td>46.6</td>
<td>68.0</td>
</tr>
</tbody>
</table>
Moddrop results

Classification accuracy on the validation set (dynamic poses)

<table>
<thead>
<tr>
<th>Modalities</th>
<th>Dropout (%)</th>
<th>Dropout + Moddrop (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>96.77</td>
<td>96.81</td>
</tr>
<tr>
<td>Mocap missing</td>
<td>38.41</td>
<td>92.82</td>
</tr>
<tr>
<td>Audio missing</td>
<td>84.10</td>
<td>92.59</td>
</tr>
<tr>
<td>Hands missing</td>
<td>53.13</td>
<td>73.28</td>
</tr>
</tbody>
</table>

Jacquard index on test set (full gestures)

<table>
<thead>
<tr>
<th>Modalities</th>
<th>Dropout (%)</th>
<th>Dropout + Moddrop (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>87.6</td>
<td><strong>88.1</strong></td>
</tr>
<tr>
<td>Mocap missing</td>
<td>30.6</td>
<td>85.9</td>
</tr>
<tr>
<td>Audio missing</td>
<td>78.9</td>
<td>85.4</td>
</tr>
<tr>
<td>Hands missing</td>
<td>46.6</td>
<td>68.0</td>
</tr>
</tbody>
</table>

Small advantage as regularizer compared to Dropout
### Moddrop results

#### Classification accuracy on the validation set (dynamic poses)

<table>
<thead>
<tr>
<th>Modalities</th>
<th>Dropout (%)</th>
<th>Dropout + Moddrop (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>96.77</td>
<td>96.81</td>
</tr>
<tr>
<td>Mocap missing</td>
<td>38.41</td>
<td>92.82</td>
</tr>
<tr>
<td>Audio missing</td>
<td>84.10</td>
<td>92.59</td>
</tr>
<tr>
<td>Hands missing</td>
<td>53.13</td>
<td>73.28</td>
</tr>
</tbody>
</table>

#### Jacquard index on test set (full gestures)

<table>
<thead>
<tr>
<th>Modalities</th>
<th>Dropout (%)</th>
<th>Dropout + Moddrop (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>87.6</td>
<td><strong>88.1</strong></td>
</tr>
<tr>
<td>Mocap missing</td>
<td>30.6</td>
<td>85.9</td>
</tr>
<tr>
<td>Audio missing</td>
<td>78.9</td>
<td>85.4</td>
</tr>
<tr>
<td>Hands missing</td>
<td>46.6</td>
<td>68.0</td>
</tr>
</tbody>
</table>

Large advantage when inputs are unreliable
Learning fusion architectures
Early vs. late fusion

**Early Fusion**
Fuse modalities at input (or preprocessed feature) level

**Late Fusion**
Fuse modalities at output level

- RGB
- Depth
- Mocap
- Audio
Deep fusion: flexibility

Path V1:
- depth video, right hand
- intensity video, right hand

Path V2:
- depth video, left hand
- intensity video, left hand

Path M:
- mocap stream

Path A:
- audio stream

max pooling

ConvD1

ConvC1

ConvD2

HLV1

HLV2

shared hidden layer HLS

output layer

ConvD1

ConvC1

ConvD2

HLV1

HLV2

ConvA1

HLA1

HLA2

pose feature extractor

mocap stream

mel frequency spectrograms
Error evolution during iterative training

Graph showing error evolution across training stages for different modalities.

- **(1)** intensity
- **(2)** depth
- **(3)** video, one hand
- **(4)** video, two hands
- **(5)** mocap
- **(6)** visual
- **(7)** audio
- **(8)** all, one scale
- **(9)** all, multiscale

Legend:
- step=2
- step=3
- step=4

Training stage vs. validation error percentage.
Criticism of our previous approach

- Emphasis on *learned* feature representations, but:
  - **Process** by which modalities were fused was designed by intuition

- Can we learn not only the features, but also the fusion process?
Fusion structure

Several aspects of fusion may be learned, for example:

• **Order** in which modalities are fused
• **Depth** learned from each modality pre-fusion
• **Amount of training** prior to fusion
• **Type of training** for each layer
• **Fusion operation**
Learning architectures

- Romanticized notion of DL - end of feature engineering
- Feature engineering has decreased
- Architectures have become more complex

http://smerity.com/articles/2016/architectures_are_the_new_feature_engineering.html
# Two views on learning fusion

<table>
<thead>
<tr>
<th></th>
<th>View #1</th>
<th>View #2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fusion strategy</strong></td>
<td>Treat each aspect of fusion as a hyperparameter</td>
<td>Propose a model framework with differentiable fusion structure</td>
</tr>
<tr>
<td><strong>Search for fusion structure</strong></td>
<td>Model-based hyperparameter optimization</td>
<td>Gradient-based optimization</td>
</tr>
<tr>
<td><strong>Efficiency to train</strong></td>
<td>Slow - must train the model to convergence to make a single update to fusion structure</td>
<td>Fast - structure and parameters can be optimized jointly</td>
</tr>
<tr>
<td><strong>Challenge</strong></td>
<td>Easy - does not require objective to be differentiable w.r.t. fusion parameters</td>
<td>Difficult - requires ingenuity in design</td>
</tr>
</tbody>
</table>
View #1 - Hyperparameter search

• Neural networks have many associated architectural and learning settings, we call these “hyper-parameters”, e.g.
  - Number of layers
  - Number of hidden units in each layer
  - Learning rate
  - Regularization (e.g. weight decay)
  - When to stop training (overstopping)

• Also the fusion structure. But how to set these?
Hyper-parameter optimization
Hyper-parameter optimization

- Traditionally, hyper-parameters have been set by:
  - expert knowledge (experience)
  - systematic search, e.g. grid, random
Hyper-parameter optimization

- Traditionally, hyper-parameters have been set by:
  - expert knowledge (experience)
  - systematic search, e.g. grid, random
- A number of approaches have been proposed recently for auto-tuning models based on Sequential Model-Based Global Optimization strategies, e.g. Bayesian Optimization

Grid search

Learning rate

10^{-5} 10^{-4} 10^{-3} 10^{-2} 10^{-1}

# Hidden units

256 1024 4096

\times

Learning rate

# Hidden units

=
Grid search

Learning rate

\[10^{-5} \times 10^{-4} \times 10^{-3} \times 10^{-2} \times 10^{-1}\]

# Hidden units

\[256 \times 1024 \times 4096\]

\[=\]

Learning rate

\[2^{-5} \times 2^{-4} \times 2^{-3} \times 2^{-2} \times 2^{-1}\]

L2 regularization

\[\times\]
Random search

Grid Layout

Random Layout

Image Credit: Bergstra and Bengio (2012)
Bayesian optimization

![Diagram showing Bayesian optimization process](Image Credit: Brochu et al. (2010))

- **t=2**
  - Observation (x)
  - Objective fn (f(·))
  - Acquisition fn (u(·))
  - Acquisition max

Bayesian optimization uses an acquisition function to determine the next location to sample. This technique balances exploration (where the objective function is very uncertain) and exploitation (trying values where the objective function is expected to be high). It aims to minimize the number of objective function evaluations and is likely to perform well in settings with multiple local maxima.
Bayesian optimization

Figure 1: An example of using Bayesian optimization on a toy 1D design problem. The figures show a Gaussian process—GP—approximation of the objective function over four iterations of sampled values of the objective function. The figure also shows the acquisition function in the lower shaded plots. The acquisition is high where the GP predicts a high objective—exploitation—and where the prediction uncertainty is high—exploration—areas with both attributes are sampled first. Note that the area on the far left remains unsampled, as while it has high uncertainty, it is—correctly—predicted to offer little improvement over the highest observation.

The posterior captures our updated beliefs about the unknown objective function. One may also interpret this step of Bayesian optimization as estimating the objective function with a surrogate function, also called a response surface, described formally in §2.1 with the posterior mean function of a Gaussian process.

To sample efficiently, Bayesian optimization uses an acquisition function to determine the next location $x_{t+1}$ to sample. The decision represents an automatic trade-off between exploration—where the objective function is very uncertain—and exploitation—trying values of $x$ where the objective function is expected to be high. This optimization technique has the nice property that it aims to minimize the number of objective function evaluations. Moreover, it is likely to do well even in settings where the objective function has multiple local maxima.

Image Credit: Brochu et al. (2010)
Bayesian optimization

Bayesian optimization uses an acquisition function to efficiently sample the objective function. The acquisition function is high where the GP predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration). The decision represents an acquisition max. The posterior captures our updated beliefs about the unknown objective function.

Figure 1: An example of using Bayesian optimization on a toy 1D design problem. The figures show a Gaussian process (GP) approximation of the objective function over four iterations of sampled values of the objective function. The figure also shows the acquisition function in the lower shaded plots. The acquisition is high where the GP predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration). The area on the far left remains unsampled, as while it has high uncertainty, it is —correctly —exploration— sampled first. Note that the area on the far left remains unsampled, as while it has high uncertainty, it is —correctly —predicted —exploitation —acquisition function.

Image Credit: Brochu et al. (2010)
Requirement #1 - Search space

- Fix the modality-specific “front end”
- n-ary fusion followed by 0-4 fully connected layers
- FC layers are constant width
Different fusion strategies lead to different accuracy

- Cornell Activity Dataset (CAD-60)
- 5 modalities:
  - Skeletal
  - Simple-HOG (Depth, Intensity)
  - Skeletal-HOG (Depth, Intensity)
- Exhaustive search

*Figure 1: Distribution of cross validation accuracies over the top network structures using Cornell C-60 dataset.*
Requirement #2 - Kernel

- Create undirected graph where nodes are architectures and edges represent mutations
- Graph-induced kernel is radial with respect to the geodesic metric on the undirected graph

\[ k(x, y) = r(d(x, y)) \]
Visualizing the kernel

Figure 2: CV accuracies against our graph kernel's distances. The simplest tree was chosen as a root of the graph and the distance to other nodes represented as the cumulative edge cost is plotted on the abscissa and the absolute difference in accuracies between those nodes is represented on the ordinate axis.

Figure 3: Mean and confidence bounds for the plot above. The data has a very positive trend but is highly heteroscedastic, which might imply that there exists a better way of assigning the scores.
Comparison to random search

- CAD-60 Dataset
- Plot shows error of the best architecture found so far (mean over 6 runs)
- All FC-layers added have 128 units
Montalbano (Chalearn)

- Less variance than CAD-60 in terms of test accuracies over different architectures
- Both BO and random search quickly find a reasonable architecture
- Importantly, we find a marginally better architecture than the hand-designed one in (Neverova et al. 2015) - in 12 iterations
## Ongoing work (View #2)

<table>
<thead>
<tr>
<th>Stochastic Regularization Method</th>
<th>Implicit Weight Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropout (Srivastava et al. 2012)</td>
<td><img src="image1.png" alt="Diagram" /></td>
</tr>
<tr>
<td>DropConnect (Wan et al. 2013)</td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Blockout (Murdoch et al. 2016)</td>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

*Image Credit: Murdoch et al. (2016)*

---

Figure 3: A summary of the types of basic high-level architectures using an implicit average with weights determined by separate cluster probabilities. In general, Dropout can be seen as another example of stochastic regularization.

During inference, we take an approach similar to dropout where the Dropout probability determines the weight given to each architecture and Blockout give block-structured, low-rank masks up to a per-layer basis. During training, we show how separate cluster probabilities can be learned for each node. Structured noise is introduced by randomly setting parameter values to zero. Similar to Dropout works by setting node activations to zero and zero otherwise: \( p \cdot w + (1 - p) \cdot 0 \rightarrow p \cdot w \) for each node. This encodes the desired behavior that a parameter be nonzero only if its corresponding input and output nodes belong to the same class while restricting that the mask be between zero and one. Let \( I \) be a binary in-stance set in Equation that randomly sets activations to zero, and DropConnect. These constraints act as a regularizer that enforces the parameter matrices to be block-structured with potentially different hierarchical architectures at each training iteration.

We first consider the case of a single fixed probability where the block-structured parameter matrices are constructed as Bernoulli random variables and draw a different hierarchical architecture at each training iteration. For each node, \( W \) indicates the element-wise multiplication between zero and one. Let \( W_j \) be a binary instance of hierarchical architectures can be summarized by the constraint set in Equation:

\[
\begin{align*}
0 & \leq I_{j,k}^t \leq 1 \\
\sum_k I_{j,k}^t & = 1
\end{align*}
\]

We instead take an approach akin to stochastic regularization to learn the hierarchical structure during training, which is equivalent to learning the cluster membership assignments as Bernoulli random variables and draw a different hierarchical architecture at each training iteration.
Blockout: Architecture by masking

- By stochastically assigning units to clusters, Blockout (Murdoch et al. 2016) permit a “library” of architectural components

![Diagram showing various architectures: Parallel, Dropout, Dropout, Split, Merge](Image Credit: Murdoch et al. (2016))

- These are the same kinds of architectural components needed to assemble multi-modal fusion architectures!
Conclusions

- Careful initialization, gradually permit more complexity or “effective parameters”
- Fuse modalities gradually
- “Moddrop” increases robustness to noise
- Bayesian Optimization is promising, but
  - Ideally we would like to learn other hyper-parameters along with the fusion strategy
Acknowledgements

- All of the work here is collaborative:

  - Natalia Neverova (Facebook AI Research)
  
  - Christian Wolf (INSA-Lyon)
  
  - Dhanesh Ramachandram and Michal Lisicki (University of Guelph)
  
  - Timothy Shields and Mohamed Amer (SRI)
Recent Github projects

Baseline Theano/Lasagne-based implementation of this work

**nneverova / deepgestures_lasagne**

- Issues: 0
- Pull requests: 0
- Wiki
- Pulse
- Graphs

No description or website provided.

- 6 commits
- 1 branch
- 0 releases
- 1 contributor

Branch: master

Easy multi-node multi-GPU parallelization of Theano models

**uoguelph-mlrg / Theano-MPI**

- Issues: 5
- Pull requests: 0
- Wiki
- Pulse
- Graphs
- Settings

MPI Parallel framework for training deep learning models built in Theano — Edit

- 135 commits
- 1 branch
- 0 releases
- 2 contributors

Branch: master

Clone or download