Unsupervised Deep Learning with Stacked Autoencoders on Chameleon

Dr. Pablo Rivas*  Ezequiel Rivas^  Deep Dand*

* Marist College, School of Computer Science and Mathematics
^ Nogales Institute of Technology, Research and Grad. Studies
The Problem

- American Sign Language (ASL)
Existing Solutions

- **Learning** the American Sign Language (ASL)
Existing Solutions

- Learning the American Sign Language (ASL)
Existing Solutions

- **Learning** the American Sign Language (ASL)
### Existing Solutions

<table>
<thead>
<tr>
<th>Method</th>
<th>Class type</th>
<th># of class</th>
<th># of subj.</th>
<th>Test w/ diff.</th>
<th>Input</th>
<th>Accur. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nagi et al. [8]</td>
<td>Gesture</td>
<td>6</td>
<td>-</td>
<td>No</td>
<td>Color</td>
<td>96</td>
</tr>
<tr>
<td>Van den Bergh et al. [14]</td>
<td>Gesture</td>
<td>6</td>
<td>-</td>
<td>No</td>
<td>Color &amp; Depth</td>
<td>99.54</td>
</tr>
<tr>
<td>Isaacs et al. [3]</td>
<td>Alphabets</td>
<td>24</td>
<td>-</td>
<td>-</td>
<td>Color</td>
<td>99.9</td>
</tr>
<tr>
<td>Pugeault et al. [10]</td>
<td>Alphabets</td>
<td>24</td>
<td>5</td>
<td>-</td>
<td>Color &amp; Depth</td>
<td>73</td>
</tr>
<tr>
<td>Pugeault et al. [10]</td>
<td>Alphabets</td>
<td>24</td>
<td>5</td>
<td>-</td>
<td>Depth</td>
<td>69</td>
</tr>
<tr>
<td>Pugeault et al. [10]</td>
<td>Alphabets</td>
<td>24</td>
<td>5</td>
<td>-</td>
<td>Color &amp; Depth</td>
<td>75</td>
</tr>
<tr>
<td>Kuznetsova et al. [6] (50/50)%</td>
<td>Alphabets</td>
<td>24</td>
<td>5</td>
<td>No</td>
<td>Depth</td>
<td>87</td>
</tr>
<tr>
<td>Kuznetsova et al. [6] (4/1)</td>
<td>Alphabets</td>
<td>24</td>
<td>5</td>
<td>Yes</td>
<td>Depth</td>
<td>57</td>
</tr>
<tr>
<td>Dong et al. [2] (50/50)%</td>
<td>Alphabets</td>
<td>24</td>
<td>5</td>
<td>No</td>
<td>Depth</td>
<td>90</td>
</tr>
<tr>
<td>Dong et al. [2] (4/1)</td>
<td>Alphabets</td>
<td>24</td>
<td>5</td>
<td>Yes</td>
<td>Depth</td>
<td>70</td>
</tr>
<tr>
<td>Ours (re-training) (50/25/25)%</td>
<td>Alph. &amp; Digit</td>
<td>31</td>
<td>5</td>
<td>No</td>
<td>Depth</td>
<td>99.99</td>
</tr>
<tr>
<td>Ours (re-training) (3/1/1)</td>
<td>Alph. &amp; Digit</td>
<td>31</td>
<td>5</td>
<td>Yes</td>
<td>Depth</td>
<td>75.18</td>
</tr>
<tr>
<td>Ours (re-training) (4/1)</td>
<td>Alph. &amp; Digit</td>
<td>31</td>
<td>5</td>
<td>Yes</td>
<td>Depth</td>
<td>78.39</td>
</tr>
<tr>
<td>Ours (fine-tuning) (3/1/1)</td>
<td>Alph. &amp; Digit</td>
<td>31</td>
<td>5</td>
<td>Yes</td>
<td>Depth</td>
<td>83.58</td>
</tr>
<tr>
<td>Ours (fine-tuning) (4/1)</td>
<td>Alph. &amp; Digit</td>
<td>31</td>
<td>5</td>
<td>Yes</td>
<td>Depth</td>
<td>85.49</td>
</tr>
</tbody>
</table>

Existing Solutions

● Learning the American Sign Language (ASL) with CNNs
The Proposal

- Learning the American Sign Language (ASL) with Auto-encoders
  - Simpler than CNN
  - More efficient than CNN
  - Faster to train than CNN
  - Similar performance to a CNN
  - CNNs are not the panacea in pattern recognition on images or computer vision
The Proposal

- *Let's talk about Auto-encoders*
The Proposal

- *Let's talk about Auto-encoders*

\[
L = \frac{1}{N} \left\| x_n - \hat{x}_n \right\|^2 + \theta_w \frac{1}{2} \sum_{l=1}^{L} \left\| w^l \right\|^2 + \theta_s \sum_{m=1}^{M} KL (\theta_\alpha \| \bar{a}_m )
\]
The Proposal

- **Let's talk about Auto-encoders**

\[
L = \frac{1}{N} \| \mathbf{x}_n - \hat{\mathbf{x}}_n \|^2_2 + \theta_w \frac{1}{2} \sum_{l=1}^{L} \| \mathbf{w}^l \|^2_2 + \theta_s \sum_{m=1}^{M} KL (\theta_\alpha \| \bar{\alpha}_m )
\]

\[
\sum_{m=1}^{M} KL (\theta_\alpha \| \bar{\alpha}_m ) = \sum_{m=1}^{M} \theta_\alpha \log \left( \frac{\theta_\alpha}{\bar{\alpha}_m} \right) + (1 - \theta_\alpha) \log \left( \frac{1 - \theta_\alpha}{1 - \bar{\alpha}_m} \right)
\]

\[
\bar{\alpha}_m = \frac{1}{N} \sum_{n=1}^{N} \psi \left( \mathbf{w}_m^{(l)T} \mathbf{x}_n + b_m^{(l)} \right)
\]
output $d_n \in \mathbb{R}_+^{31}$

softmax activations

$\tilde{x}_n \in \mathbb{R}_+^{50}$

logistic activations

$\hat{x}_n \in \mathbb{R}_+^{100}$

sparse encoding layer

$logistic$ activations

$x_n \in \mathbb{N}_+^{65536}$

sparse encoding layer

$E = \frac{1}{N} \sum_{n=1}^{N} \sum_{c \in C} \hat{d}_{cn} \ln d_{cn} + (1 - \hat{d}_{cn}) \ln(1 - d_{cn})$

neural network layer

input image
Results

- of the Auto-encoders

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>0.9748</td>
<td>0.9923</td>
<td>0.9935</td>
<td>0.9929</td>
<td>0.9910</td>
<td>0.9889</td>
</tr>
<tr>
<td>SPC</td>
<td>0.9991</td>
<td>0.9997</td>
<td>0.9998</td>
<td>0.9998</td>
<td>0.9997</td>
<td>0.9996</td>
</tr>
<tr>
<td>MAE</td>
<td>0.1483</td>
<td>0.0640</td>
<td>0.0373</td>
<td>0.0347</td>
<td>0.0494</td>
<td>0.0667</td>
</tr>
</tbody>
</table>
Results

The chart illustrates the number of total errors for different target classes. Bars represent the incorrectly predicted classes.
Results
Results
Chameleon Setup

- **Resource type:** bare metal/CHI@TACC
- **Lease:** GPU P100
- **Image:** CC-Ubuntu16.04-CUDA8
- **Libraries:**
  - cuDNN
  - libatlas-dev
Libraries: cuDNN
Chameleon Setup

- **Resource type:** bare metal/CHI@TACC
- **Lease:** GPU P100
- **Image:** CC-Ubuntu16.04-CUDA8
- **Packages:**
  - gcc, gfortran
  - python-\{numpy scipy matplotlib\}
  - tensorflow
  - glances, nvidia-ml-py, screen
Packages: tensorflow

11.0)
Downloading funcsigns-1.0.2-py2.py3-none-any.whl
Collecting pbr>=0.11 (from mock>=2.0.0->tensorflow==0.11.0)
  Downloading pbr-3.1.1-py2.py3-none-any.whl (99kB)
    100% |████████████████████████████████| 102kB 11.7MB/s
Requirement already up-to-date: setuptools in /usr/local/lib/python2.7/dist-packages (from protobuf==3.0.0->tensorflow==0.11.0)
Installing collected packages: funcsigns, pbr, mock, protobuf, tensorflow
  Found existing installation: funcsigns 0.4
  Uninstalling funcsigns-0.4:
    Successfully uninstalled funcsigns-0.4
  Found existing installation: pbr 1.8.0
  Uninstalling pbr-1.8.0:
    Successfully uninstalled pbr-1.8.0
  Found existing installation: mock 1.3.0
  Uninstalling mock-1.3.0:
    Successfully uninstalled mock-1.3.0
Successfully installed funcsigns-1.0.2 mock-2.0.0 pbr-3.1.1 protobuf-3.0.0 tensorflow-0.11.0
cc@fox:~$
cc@fox:~:/research/code-grid-search/summaries$ tensorboard --logdir=.
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcudnn.so locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcudade.so.1.0.0 locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcudarender.so.1 locally
Starting TensorBoard 29 on port 6006
(You can navigate to http://127.0.0.1:6006)
Packages: glances

---

Uptime: 2:38:42

**CPU**
- User: 1.4%
- Proc: 1.4%

**MEM**
- System: 0.7%
- Mem: 96% used: 4.45G

**SWAP**
- Idle: 97.9%

**NETWORK**
- Rx/s: 4Kb
- Tx/s: 784b

**TASKS**
- 458 tasks (581 threads), 2 run, 456 sleep, 0 others

**DISK I/O**
- R/s: 0, W/s: 912K

**FILE SYS**
- / (sda1): 7.92G used, 218G total

2017-09-12 19:19:26

No warning or critical alert detected
No warning or critical alert detected

TASKS 459 (587 thr*), 2 run, 457 slp, 0 oth sorted automatically by cpu_percent, flat view
<table>
<thead>
<tr>
<th>CPU</th>
<th>61.2%</th>
<th>GPU Tesla M40</th>
<th>MEM</th>
<th>5.0%</th>
<th>SWAP</th>
<th>0.0%</th>
</tr>
</thead>
<tbody>
<tr>
<td>user:</td>
<td>54.3%</td>
<td>proc:</td>
<td>39%</td>
<td>total:</td>
<td>126G</td>
<td>total:</td>
</tr>
<tr>
<td>system:</td>
<td>16.3%</td>
<td>mem:</td>
<td>96%</td>
<td>used:</td>
<td>6.24G</td>
<td>used:</td>
</tr>
<tr>
<td>idle:</td>
<td>29.4%</td>
<td></td>
<td>free:</td>
<td>119G</td>
<td>free:</td>
<td>0</td>
</tr>
</tbody>
</table>

**NETWORK**

<table>
<thead>
<tr>
<th>eno1</th>
<th>Rx/s</th>
<th>Tx/s</th>
<th>240b</th>
<th>1Kb</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo</td>
<td>176b</td>
<td>176b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DISK I/O**

<table>
<thead>
<tr>
<th>sda1</th>
<th>R/s</th>
<th>W/s</th>
<th>7K</th>
<th>0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>sr0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FILE SYS**

<table>
<thead>
<tr>
<th>/ (sda1)</th>
<th>Used</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.58G</td>
<td>218G</td>
<td>0.0</td>
</tr>
</tbody>
</table>

2017-09-12 20:02:25  No warning or critical alert detected
### CPU Usage

| CPU  | User: 43.9% | Nice: 0.0% | Context: 0x0000000000000000 | GPU Tesla M40 | MEM: 5.1% Active: 5.5G | SWAP: 0.0% | LOAD: 48-core |

### Memory Usage


### Network

- **Rx/s** 1.6Gbps, **Tx/s** 1.6Gbps

### Disk I/O

<table>
<thead>
<tr>
<th>Device</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo</td>
<td>1.0G</td>
<td>1.0G</td>
</tr>
</tbody>
</table>

### CPU Usage by Processes

<table>
<thead>
<tr>
<th>Process</th>
<th>CPU Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>3246.9</td>
<td>1.7%</td>
</tr>
<tr>
<td>10.8G</td>
<td>2.10G</td>
</tr>
<tr>
<td>31154</td>
<td>101.1G</td>
</tr>
</tbody>
</table>

### File Systems

- **Total Size**: 218G
- **Used Size**: 1.9G

### Sensors

- **Physical id 0**: 45°C
- **Core 1**: 39°C
- **Core 2**: 43°C
- **Core 10**: 46°C
- **Core 11**: 39°C
- **Core 13**: 36°C
- **Core 0**: 40°C
- **Core 1**: 35°C
- **Physical id 1**: 36°C

### System Status

- No warning or critical alert detected.
Conclusions

- Learning the American Sign Language (ASL) with Auto-encoders
  - Simpler than CNN
  - More efficient than CNN
  - Faster to train than CNN
  - Similar performance to a CNN
  - CNNs are not the panacea in pattern recognition on images or computer vision (no free lunch theorem)
Unsupervised Deep Learning with Stacked Autoencoders on Chameleon

Interested in code? Check Deep’s repo:

https://github.com/DeepDand/research