# An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices

1. Zeenat Tariq
2. Sayed Khushal Shah

UMKC

# Discussion Agenda

1. Introduction
2. Study Preliminaries
3. Performance Benchmark
4. Layer and Unit Profiling
5. Model Footprint
6. Conclusion
7. References

## Introduction

- An initial measurement study designed to provide critical first order insights in the development of embedded and mobile device support for deep learning
- Specially, the systematic profiling of the two most commonly used deep learning model architectures
  - Deep Neural Network
  - Convolutional Neural Network
- Experiments are conducted using three hardware platforms (e.g. IoT, wearable and mobile applications)

## Continued …

- First, Whole model benchmarks are performed for all platforms to better understand what is currently feasible, and importantly how energy consumption and execution time corresponds to existing application requirements
- Second, the relationship with resource usage and internal model architecture is studied as it relates to architecture design and related algorithmic choices
- Third, and finally: memory and model footprint needs are investigated, especially within the context of hardware capabilities

# 2. STUDY PRELIMINARIES

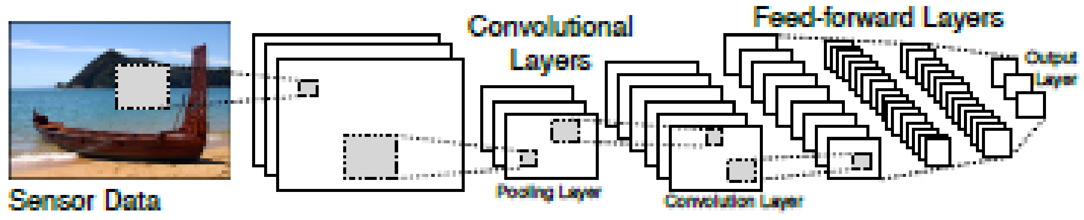
**Deep Neural Network:**

- In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer’s output
- The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer

# Continued..

**Convolutional Neural Network:**

- CNNs are an alternative to DNNs that still share many architectural similarities. A CNN is composed of one or more convolutional layers, pooling or sub-sampling layers, and fully connected layers.
- The aim of these layers is to extract simple representations at high resolution from the input data, and then converting these into more complex representations, but at much coarser resolutions within subsequent layers



## Continued..

(a) Snapdragon 800

(b) Tegra K1

(c) Edison

| | Type | Size | Architecture |
|---|---|---|---|
| AlexNet | CNN | 60.9M | $c$:5$^{\S}$; $p$:3$^{\ddagger}$; $h$:2$^{\star}$; $n$:{all 4096}$^{\dagger}$ |
| SVHN | CNN | 313K | $c$:2$^{\S}$; $p$:2$^{\ddagger}$; $h$:2$^{\star}$; $n$:{1600,128}$^{\dagger}$ |
| Deep KWS | DNN | 241K | $h$:3$^{\star}$; $n$:{all 128}$^{\dagger}$ |
| DeepEar | DNN | 2.3M | $h$:3$^{\star}$; $n$:{all 512 or 256}$^{\dagger}$ |

$^{\S}$*convolution layers*; $^{\ddagger}$*pooling layers*; $^{\star}$*hidden layers*; $^{\dagger}$*hidden nodes*

# Deep Model Representation

- **AlexNet**: This object recognition model supports more than 1,000 object classes (e.g., dog, car), and is by far the most complex model studied (60.9M parameters, the next highest is just 2.3M). In 2012, it offered state-of-the-art levels of accuracy for well-known datasets like ImageNet.

- **SVHN:** Specializing in extracting numbers from complex and noisy scenes, this model  has been used to recognize house numbers from Google Street view images.

- **Deep KWS:** Designed for resource constrained devices, this audio model recognizes 22 spoken words. It targets use cases like phones reacting to specific phrases (Hey Siri), an inference task often called keyword spotting (KWS).

- **DeepEar:** This also is an audio model designed for constrained use. Unusually, it is a composite of 3 coupled DNNs offering separate recognition tasks (viz. emotion recognition, speaker identification and ambient sound classification).

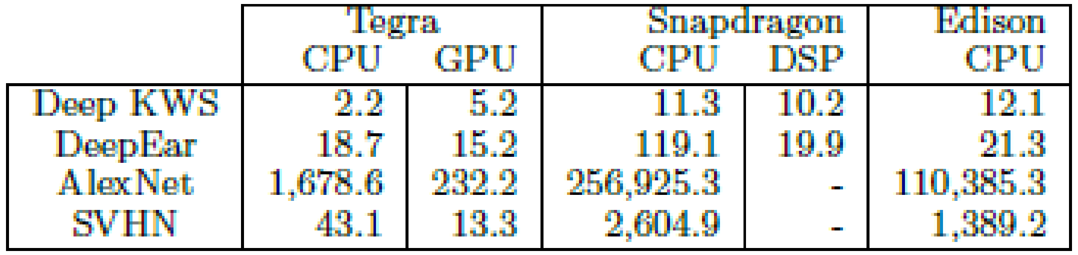
## 3. Performance Benchmark

- The first set of experiments are designed to test the raw feasibility of executing deep learning models across target hardware platforms

- The paper also consider end-to-end model performance metrics of execution time and energy consumption, especially in terms of how these map to application needs

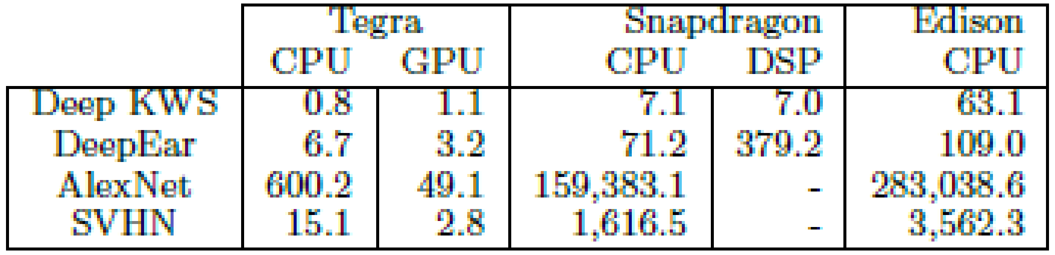
# Experimental and Performance

- Execution time and Energy Consumption



| | Tegra | | Snapdragon | | Edison |
|---|---|---|---|---|---|
| | CPU | GPU | CPU | DSP | CPU |
| Deep KWS | 2.2 | 5.2 | 11.3 | 10.2 | 12.1 |
| DeepEar | 18.7 | 15.2 | 119.1 | 19.9 | 21.3 |
| AlexNet | 1,678.6 | 232.2 | 256,925.3 | - | 110,385.3 |
| SVHN | 43.1 | 13.3 | 2,604.9 | - | 1,389.2 |

**Table 2:** Energy Consumption (mJ.)



| | Tegra | | Snapdragon | | Edison |
|---|---|---|---|---|---|
| | CPU | GPU | CPU | DSP | CPU |
| Deep KWS | 0.8 | 1.1 | 7.1 | 7.0 | 63.1 |
| DeepEar | 6.7 | 3.2 | 71.2 | 379.2 | 109.0 |
| AlexNet | 600.2 | 49.1 | 159,383.1 | - | 283,038.6 |
| SVHN | 15.1 | 2.8 | 1,616.5 | - | 3,562.3 |

**Table 3:** Execution Time (msec.)

## Continued..

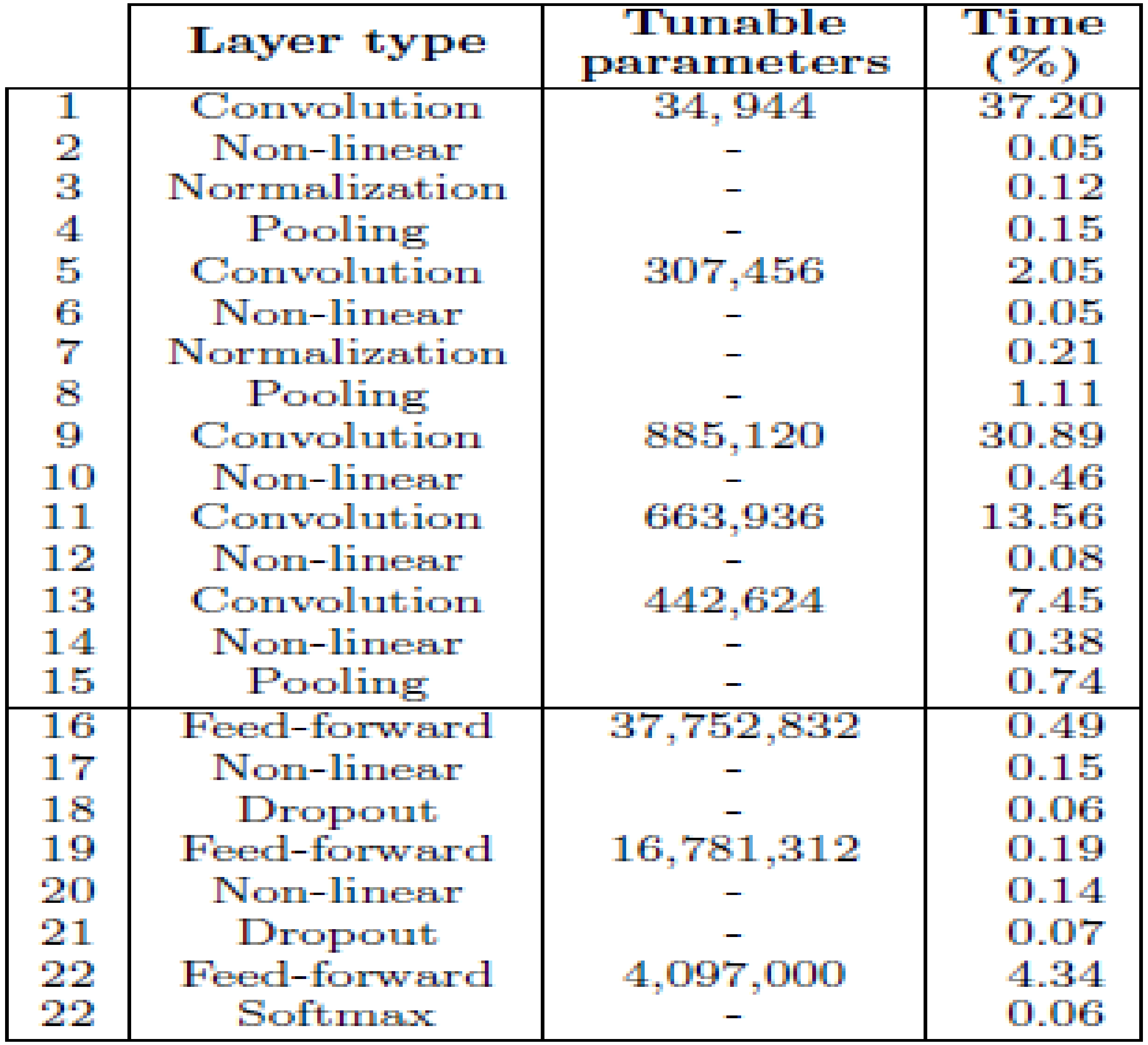
- Deep KWS have acceptable performance with execution times ranging between 63 and <1 msec
- Deep Ear is an exception to this tendency for instance, although it has 7 more parameters than SVHN it is also on average is 22 faster in execution time
- The Tegra GPU is by far the most efficient processor for CNN layers while DNN layers have a more uniform processor performance

## 4. LAYER AND UNIT PROFILING

- The paper next examine factors contributing to the computation related overhead observed
- Computational load is tied closely with other performance metrics including execution time and energy efficiency.
- The focus in particular on the consequences of architecture (e.g. layer type) and algorithmic choices (e.g. activation function)

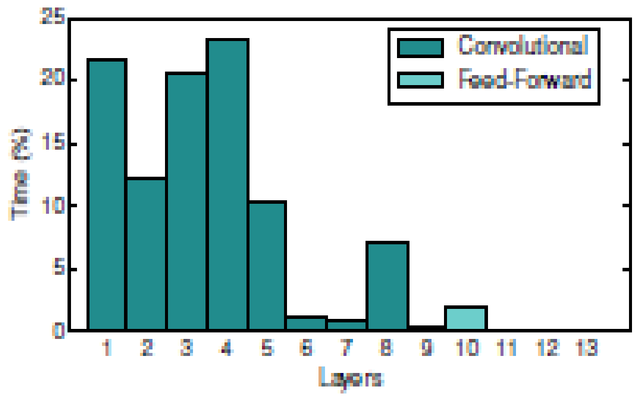
## Experimental Set Up

- Through code annotation the execution time of individual layers and units is profiled
- The paper highlight’s performance characteristics not tied to platform differences, and emphasize this by reporting the percentage of execution time (for unit, or layer) as the primary metric used
- Similarly, all reported effects are based on measurements from a single platform; importantly, it is also observed the same performance trends across other test hardware platforms
- Experiments to test the sensitivity to activation functions are done by replacing the relevant layer, retraining the model, and repeating inference tests

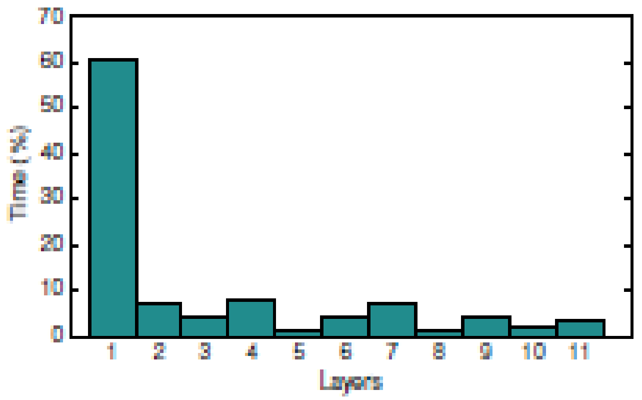


| | Layer type | Tunable parameters | Time (%) |
|---|---|---|---|
| 1 | Convolution | 34, 944 | 37.20 |
| 2 | Non-linear | - | 0.05 |
| 3 | Normalization | - | 0.12 |
| 4 | Pooling | - | 0.15 |
| 5 | Convolution | 307,456 | 2.05 |
| 6 | Non-linear | - | 0.05 |
| 7 | Normalization | - | 0.21 |
| 8 | Pooling | - | 1.11 |
| 9 | Convolution | 885,120 | 30.89 |
| 10 | Non-linear | - | 0.46 |
| 11 | Convolution | 663,936 | 13.56 |
| 12 | Non-linear | - | 0.08 |
| 13 | Convolution | 442,624 | 7.45 |
| 14 | Non-linear | - | 0.38 |
| 15 | Pooling | - | 0.74 |
| 16 | Feed-forward | 37,752,832 | 0.49 |
| 17 | Non-linear | - | 0.15 |
| 18 | Dropout | - | 0.06 |
| 19 | Feed-forward | 16,781,312 | 0.19 |
| 20 | Non-linear | - | 0.14 |
| 21 | Dropout | - | 0.07 |
| 22 | Feed-forward | 4,097,000 | 4.34 |
| 22 | Softmax | - | 0.06 |

- Layer by layer computational Overhead for one DNN and CNN on Tegra GPU

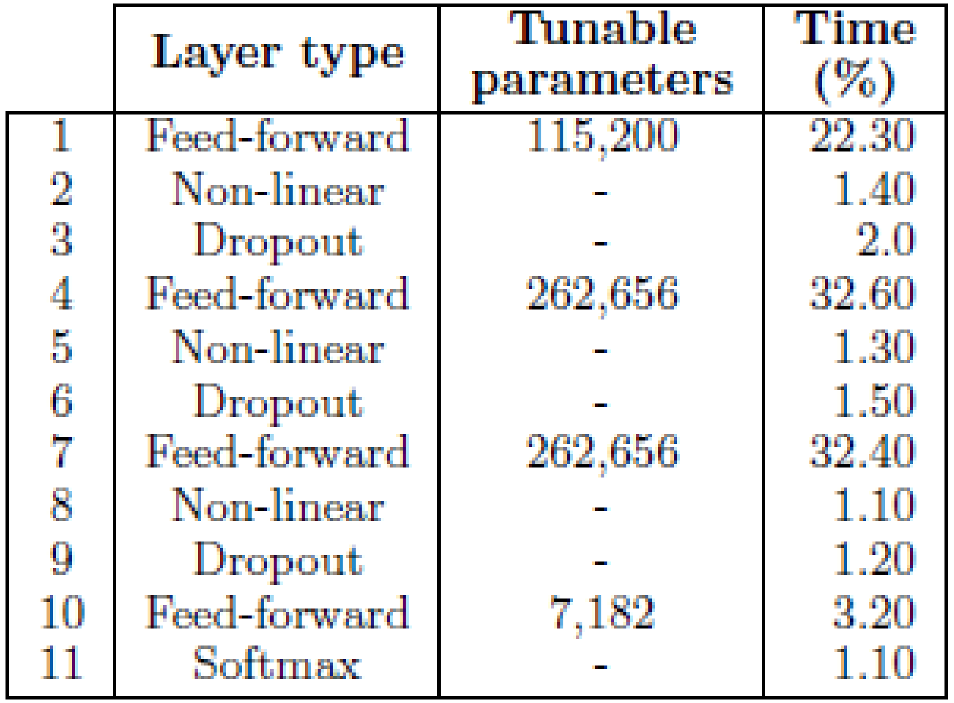


(a) SVHN



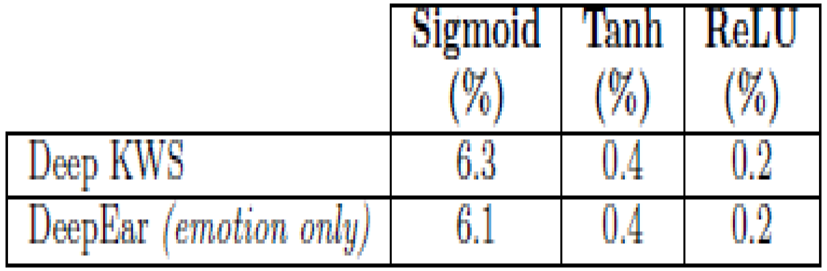
(b) Deep KWS

- Layer by layer performance of DNN with in Deep Ear that performs emotion recognition
- Inference time 30msec
- Audio frame 0.00035msec
- Execution on Tegra GPU



| | Layer type | Tunable parameters | Time (%) |
|---|---|---|---|
| 1 | Feed-forward | 115,200 | 22.30 |
| 2 | Non-linear | - | 1.40 |
| 3 | Dropout | - | 2.0 |
| 4 | Feed-forward | 262,656 | 32.60 |
| 5 | Non-linear | - | 1.30 |
| 6 | Dropout | - | 1.50 |
| 7 | Feed-forward | 262,656 | 32.40 |
| 8 | Non-linear | - | 1.10 |
| 9 | Dropout | - | 1.20 |
| 10 | Feed-forward | 7,182 | 3.20 |
| 11 | Softmax | - | 1.10 |

## Unit Analysis



|  | Sigmoid (%) | Tanh (%) | ReLU (%) |
|---|---|---|---|
| Deep KWS | 6.3 | 0.4 | 0.2 |
| DeepEar *(emotion only)* | 6.1 | 0.4 | 0.2 |

- Shows the impact of selecting one of three popular activation functions, in terms of how their computation throughout the model contributes to overall execution time for DNN-based models
- Contribution to overall runtime of non-linear layers for Deep KWS and DeepEar
- Inuence of activation function choice is shown

## 5. MODEL FOOTPRINT

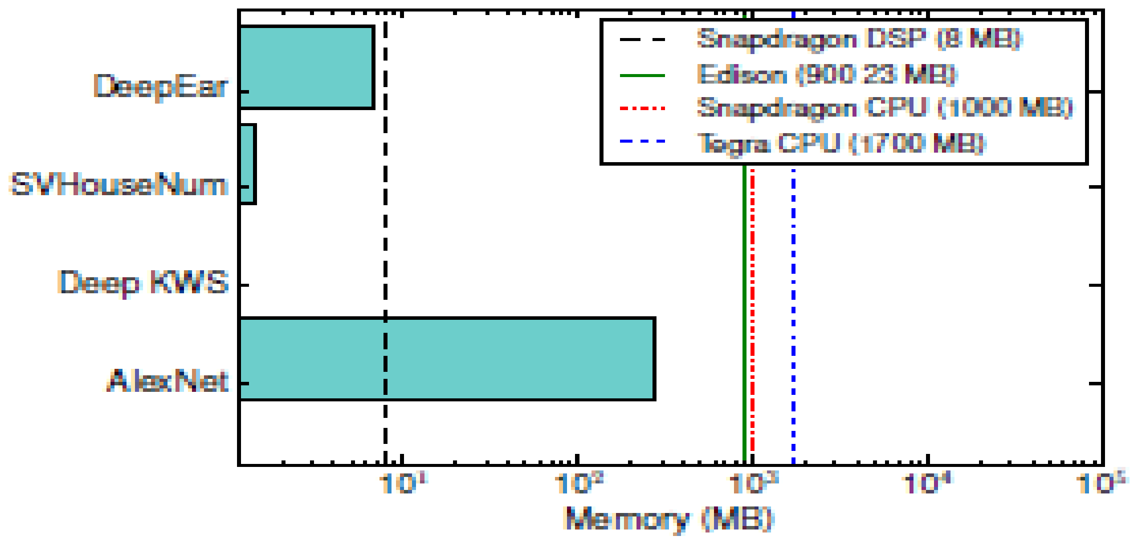
- Final experiments examine the memory footprint of each model, this is known to be a key bottleneck in the use of deep learning on resource constrained devices due to their many parameters
- Comparison of model memory requirements against availability on each hardware platform.

## Experiment

- In these experiments we determine overall, and per layer, memory usage during inference for each model
- Aim of this implementation is to carefully use memory during execution, and in its representation of models.
- By default the whole model resides in memory during processing.
- For single layer profiling this behavior is changed and only the layers being operated upon reside in memory
- Report measurements from the Snapdragon CPU, we find these generalize well across the platforms.

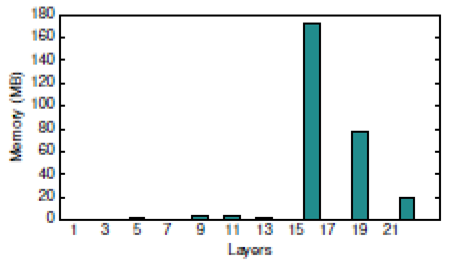
## Whole Model Requirement

- The overall DNN/CNN model size across all layers in comparison to the maximum available memory
- AlexNet occupies nearly 300 MB which is roughly a third of the memory available for the Edison and Snapdragon CPU, and is 37 times the maximum memory available to the Snapdragon DSP.

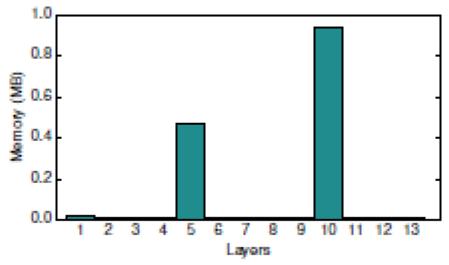


# Memory Fluctuations

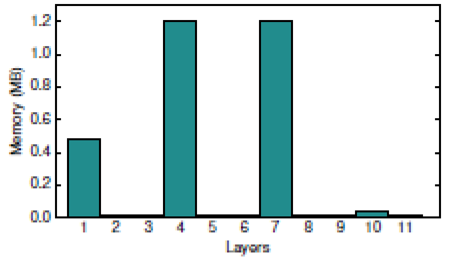
- Displays per layer memory requirements



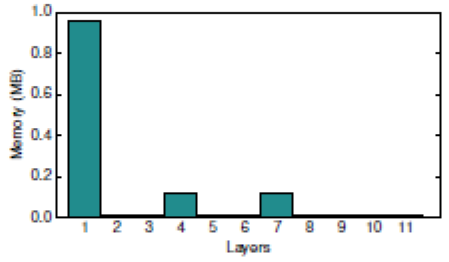
(a) AlexNet



(b) SVHN



(c) DeepEar *(emotion only)*



(d) Deep KWS

- This shows 93% of memory is only required during the processing of 3 of the 22 layers, with all 3 of the layers occurring briefly in the later part of inference
- The considerable difference between the memory needs of DNNs and CNNs
- CNNs require far less space than their DNN counterparts.

# Continued..

- This is due to convolution layers requiring far fewer parameters than those of DNNs. Similar to prior comments this implies for platforms that are restricted in memory, but have computation available, then the use of feed-forward layers might be reduced.
- One example of this is the Snapdragon DSP, which is severely memory limited but has a processor that excels at certain computations.

## 6. Conclusion and Future Work

- In this work, a preliminary measurement study was done into deep learning algorithms relevant to IoT and mobile applications.
- The aim of this study has been two fold:
  - First, to provide initial observations as to the execution performance of this type of sensor processing when deployed on mobile and IoT class hardware.
  - Second, to begin building the knowledge necessary to design general purpose solutions for both reducing and managing the resources consumed by these learning algorithms.
- It is suggested performance can improve if feed forward layers (in CNNs) are replaced with cheaper to compute shallow methods

UMKC

# References

- N. D. Lane, et al. Deepear: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments using Deep Learning. UbiComp '15.
- R. LiKamWa, et al. Energy Characterization and Optimization of Image Sensing Toward Continuous Mobile Vision. MobiSys '13.
- K. Simonyan, et al. Very Deep Convolutional Networks for Large-scale Image Recognition. ICLR '15.

Thank you