Delta Capita

Machine Learning Model Compression

Edward Adcock

Version v1.0
March 2019

Background

Machine Learning Model Compression

New machine learning techniques and deep learning architectures have led to models that have significantly higher accuracy rates over their predecessors. However, they rely on 'millions' to 'billions' of internal model parameters, which must be trained over very long periods of time.

As more complex models are developed, the desirable qualities of the deep learning models are:

- **Low Storage Requirements**
- **Computational Efficiency**
- **Fast Inference**

These qualities become critical factors when considering real-time applications, like high-frequency trading algorithms or deployment on mobile devices.

Most model compression techniques have their origin in a related field of study that aims to prevent over-fitting of deep learning networks.

The various approaches have been broadly grouped into four categories:

- **Parameter Pruning and Sharing**
- **Low-rank Factorization**
- **Transferred/Compact Convolutional Filters**
- **Knowledge Distillation**

Model compression has two main objectives; 1) reduce the space required to store and load a machine learning model and 2) reduce the inference time such that the model is quicker at producing predictions.

Many model compression methods focus on objective number 1, reducing the size of the model, however these have an impact on inference times. The opposite, however is also true. Approaches that focus on objective 2 also impact the size of models.

Approaches that focus on creating faster models and that are applied during model training have the following effects:

- An increase of training time, because models must be constantly retrained after each reduction step
- Faster inference time (once deployed) as less calculations need to be made

At this point, it is relevant to establish the difference between model compression and optimization. Compression is about making the model smaller and speed up inference. Model optimization on the other hand is about speeding up the training process that creates the model in the first place.

Summary of Categories:

Name	Description	Applications	Further details
Parameter pruning and sharing	Removing redundant internal model parameters	Deep Learning Architectures	Can support both training from scratch and pre-trained models
Low-rank factorization	Appling matrix decomposition to ascertain informative links	Deep Learning Architectures	Can support both training from scratch and pre-trained models
Transferred/compact convolutional filters	Designing special layers to save parameters	Convolutional neural networks only	Only supports training from scratch as new layers are introduced
Knowledge distillation	Training a new, smaller model to predict the large model	Deep Learning Architectures	Only support training from scratch (reduced training time)

Challenges

Compression Techniques

The main challenge present when using compression techniques, is the trade off between model size and model accuracy. This is because the model accuracy will tend to decrease as stronger and/or more techniques are applied to the network.

Parameter Pruning and Sharing

One of the main aspects of these techniques are, that they attempt to remove redundant or non-informative connections between nodes.

The way in which the technique decides which connections to remove is a significant challenge as there are different ways of measuring the information entropy of a connection.

Parameter pruning and sharing challenges:

- l_1 or l_2 normalization is usually applied at each stage which increases the time to convergence.
- All pruning requires some manual step to determine the parameters and/or layers that it is applied to, which can significantly impact the resulting accuracy
- The derivation of the reduced structural matrix is not analytically solvable and so must be found using other methods

"Model accuracy will tend to decrease as stronger techniques are applied."

"Tech support says the problem is located somewhere between the keyboard and my chair."

Low-rank Factorization

Low-rank factorization techniques use matrix/tensor decomposition to estimate the informative parameters of the deep learning networks. These techniques apply matrix decomposition algorithms, such as single value decomposition to remove the redundancy from the layer matrices. These can be used for both convolutional and fully connected layers. Low-rank approaches support both training from scratch and pre-trained models.

Low-rank approximation provides a potentially genuine way of reducing model complexity without a significant impact on the accuracy of the model. The way this is done is to reduce the rank (size) of each matrix representation for each layer.

"A genuine way of reducing model complexity without a significant impact on the accuracy of the model."

The challenges of this approach are similar to structural matrix techniques. The decomposition of each layer's matrix is computationally expensive and cannot be derived from an analytical solution and so must be found using other methods. In addition, the factorization requires the model to be constantly retrained to achieve convergence.

Transferred/Compact Convolutional Filters

These techniques can only be applied to convolutional neural networks and are required to be implemented at training time as most of these introduce new network structures/layers or significantly modify the network.

Example techniques are:

- Group Equivariant Convolutional Networks
- Doubly Convolutional Neural Networks
- CNNs With Concatenated Rectified Linear Units

Challenges

Transferred and compact convolutional filters have been shown to produce state-of-the-art accuracy with wide/flat architectures such as:

- VGGNet

But not on narrow/deep architectures such as:

- GoogleNet
- ResNet

Another challenge presents itself when applying the transfer assumptions to the convolutional layers because, sometimes, the assumptions are too strong to guide the algorithm, making the results unstable on some datasets.

"The significant challenge with these techniques revolves around the implementation of the additional layers."

The significant challenge with these techniques revolves around the implementation of the additional layers. In order to correctly change or add these new layers, the data scientist will need to be able to apply the relevant technique given the domain challenge as naive implementations reduce network accuracy and increase the size of the model.

Knowledge Distillation

Knowledge distillation is a model compression technique in which a smaller 'student' network model is trained to mimic the output of a larger pre-trained 'teacher' network model. The technique is particularly useful for classification tasks with multiple classes, such as:

- Handwritten Digit Recognition
- Document Classification
- Image Classification

The application of distilled models is limited to deep learning networks with the SoftMax function as their output layer.

They are most suitable for classification tasks with many possible classes. They are not applicable to regression problems and less appropriate for binary classification tasks.

"According to the computer, I need to back up your kidneys, defragment your liver and reboot your heart."

One of the major challenges with the standard knowledge distillation techniques, is that they do not perform very well for deep learning networks. FitNets were developed to address this, however, these can easily be subject to over regularization which is difficult to mitigate.

Mastering the Challenges

The main approach to tackling challenges in machine learning compression is to produce models which are significantly smaller in both memory and computational requirements without sacrificing the accuracy of the complete original model.

Model compression techniques reduce the computational requirements necessary to store and run a model.

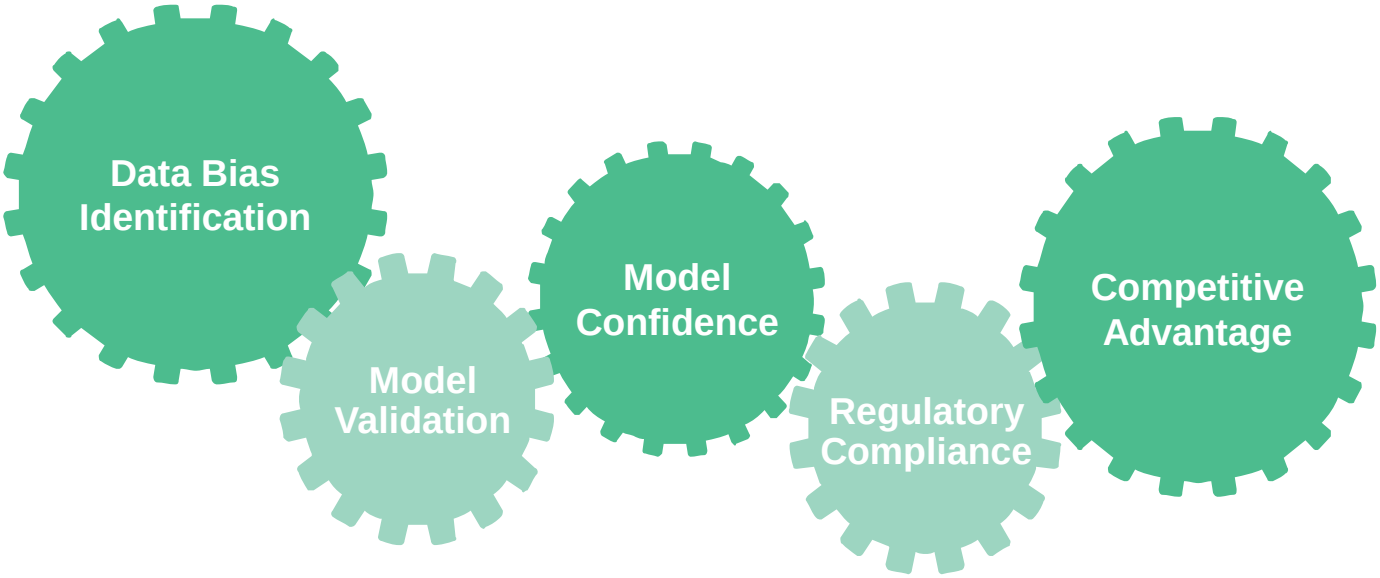
These enable a much wider deployment of deep learning models including:

- Latency sensitive applications, such as trading models, for which the ability to produce predictions within milliseconds or microseconds is important.
- Deployment on smaller devices with limited computational resources (memory and CPU). For example in contexts where the model needs to run "locally" to avoid transferring sensitive data over the wire to a remote "model server".

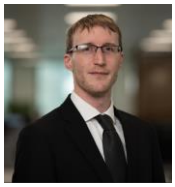
Finally, it is worth noting that each of the four types of algorithms discussed in this white-paper are orthogonal and in many ways complementary to each other, i.e. they can be combined to produce a better result.

DC MINT

The Delta Capita DC MINT platform helps our clients to address the machine learning interpretability challenges. It is a product for data scientists, validation teams, risk, compliance and other end-users.



Edward Adcock
Data Science Consultant



Data scientist and Machine Learning specialist with 14+ years experience in financial services, applying state of the art algorithms and neural network architectures on complex data sets.

Other contributors:
Sylvia Smit, Managing Partner
Ricardo Cruz, Senior Consultant
Khrystyna Andronova, Senior Consultant

Please contact Delta Capita at *capitalmarketsdelivery@deltacapita.com* if you are interested in any of the following areas:

- **DC MINT - Model Interpretability**
- **DC COMPRESSION - Model Compression and Model Optimisation**
- **DC DOCS - Automated Model Documentation**
- **DC VOICE - Voice/NLP on large data sets and Wealth applications**
- **DC RISK - Contagion Risk Models**

DC MINT