Shallow Convolutional Neural Network Architectures for Music Genre Classification

George Herbert
Department of Computer Science
University of Bristol
Bristol, United Kingdom
cj19328@bristol.ac.uk

***Abstract*—In this paper I implement and evaluate the shallow convolutional neural network architecture proposed by Schindler et al. [1] for the task of music genre categorisation.**

***Index Terms*—Music Information Retrieval, Music Genre Classification, Convolutional Neural Networks**

I. Introduction

The explosive growth of digital music platforms has sparked significant advancements in the field of music information retrieval (MIR), a rapidly evolving discipline that focuses on developing computational techniques to extract valuable insights from music and audio signals. As MIR technologies continue to advance, they are becoming increasingly crucial for the music industry, enabling the creation of more effective tools such as recommender systems, which can provide a competitive edge in a crowded market.

One of the key challenges in MIR is genre classification, which involves identifying the musical genre of a given audio signal. Accurate genre classification can help music providers organise and categorise their catalogs, and enable users to search and discover new music in a more efficient and effective way. Early efforts to solve this problem, such as that proposed by Tzanetakis and Cook [2], employed statistical classifiers that were trained on vector summaries of features such as timbral texture, rhythmic content and pitch content. However, these summaries fail to capture the temporal structure of the underlying audio. In recent years, researchers have turned to audio spectrograms, which represent frequency data over time, to train state-of-the-art deep-learning models that can effectively classify audio signals based on genre.

Convolutional neural networks (CNNs) are one type of network that have been widely employed, following their successes in the field of computer vision. In this paper, we explore the use of CNNs for genre classification, and specifically investigate the shallow CNN architecture proposed by Schindler et al. in [1], which was shown to achieve modest performance on this task.

II. Related Work

This section summarises some of the recent state-of-the-art approaches to genre classification.

Liu et al. [3] recently proposed a novel architecture named a Bottom-up Broadcast Neural Network (BBNN) to deal with some of the problems traditionally associated with genre classification. They identified that many previously developed architectures had focused on abstracting high-level semantic features layer-by-layer; as a result, these architectures suffer from a huge loss of lower-level features which are critical to the task of genre classification. Thus, the BBNN architecture was specifically designed to deal with this problem with the introduction of the novel Broadcast Module (BM), which makes use of Inception modules connected by dense connectivity. The Inception modules contain parallel convolutional and a max-pooling layers of varying shapes, thus enabling the network to capture temporal and spectral hierarchies at multiple scales. The dense connectivity enables the low-level information extracted from the earlier modules to be propagated throughout the network. Liu et al. demonstrated that the BBNN achieves state-of-the-art performance on a variety of music datasets.

One of the main problems in the wider MIR domain is the lack of the large training datasets that are frequently required to train powerful deep neural networks. To deal with this problem, Hung et al. [4] very recently published a novel method known as input-dependent neural model reprogramming (ID-NMR). ID-NMR is a transfer learning training scheme, that leverages pre-trained models from a source domain and applies it to a target domain. Hunt et al. successfully applied this technique to transform two models trained on speech and audio data to the task of genre classification. Their model managed to outperform both a fine-tuning transfer learning method and existing state-of-the-art models pre-trained on music-specific datasets.

III. Dataset

I used the GTZAN dataset, compiled by Tzanetakis and Cook [2], to train and evaluate my models. The dataset contains a total of 1000 WAV audio tracks, each 30 seconds in length. GTZAN is a balanced dataset, containing 100 tracks for each of the 10 genres labelled in the dataset: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock.

I utilised a stratified-by-genre split to produce a training and test set: the training set consisted of 750 tracks (75 tracks from each genre), while the test set contained the remaining 250 tracks.

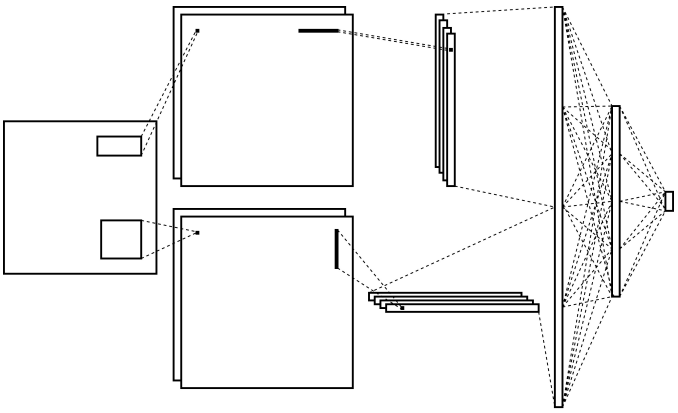


Fig. 1: Schematic representation of the shallow CNN architecture as described by Schindler et al. in [1].

IV. CNN Architecture

I recreated the shallow CNN architecture described by Schindler et al. in [1]. Figure 1 displays a schematic representation of the architecture. The network takes log-mel spectrograms of shape 80×80 as input, of which the dimensions correspond to frequency and time. To effectively process the temporal and spectral characteristics of the input spectrograms, the architecture employs a parallel design. The upper pipeline captures frequency relations in the input, while the lower pipeline captures temporal relations.

The upper pipeline includes a convolutional layer with 16 kernels of shape 10×23 (with padding), which produces 16 sqaure feature maps of shape 80×80. These feature maps are then downsampled using a max pooling layer with a window of shape 1×20, resulting in 16 vertical rectangular feature maps of shape 80×4.

The lower pipeline also includes a convolutional layer with 16 kernels (with padding), but these are approximately square of shape 21×20. The resulting 16 square feature maps of shape 80×80 are downsampled using a max pooling layer with a window of shape 20×1, resulting in 16 horizontal rectangular feature maps of shape 4×80.

The 16 feature maps from each piepline are flattened and concatenated to a shape of 1×10240, which is mapped to a 200 neuron fully-connected layer. These final 200 neurons are then mapped to 10 output neurons, with 10% dropout utilised to mitigate overfitting. The softmax function is applied to these 10 output neurons to produce a pseudo-probability distribution that indicates the probability that a given input belongs to each of the 10 genres.

Except for the final layer, each convolutional and fully-connected layer is passed through a Leaky ReLU activation function with $\alpha = 0.3$. Leaky ReLU is an extension to the ReLU activation function that outputs a small non-zero value $f(x) = \alpha x$ for negative inputs.

V. Implementation Details

A. Preprocessing

The network was trained and evaluated using a total of 15000 log-mel spectrograms. To create these spectrograms, each audio track in the GTZAN dataset was first split into chunks of approximately 0.93 seconds, using a step size of 50%. 15 randomly selected chunks from each track were then transformed into log-mel spectrograms of shape 80×80 using a fast Fourier transform with a window size of 1024, and a step size of 50%. To avoid data leakage, the tracks were split into training and test sets before the spectrograms were created. This ensured the model would not have access to any information about the test set during training, allowing for a fair and accurate evaluation of its performance.

B. Training Details

I constructed and trained the CNN using Python and the PyTorch [5] machine learning framework. The training process was implemented according to the method described by Schindler et al. in [1]. I used cross-entropy loss to evaluate the performance of the network, and employed L1 regularisation with a penalty value of 0.0001 to mitigate overfitting. I used the Adam optimiser [6]—an extension of stochastic gradient descent—to optimise the network, using a learning rate of 5×10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$ for numerical stability. I trained the network on a BlueCrystal Phase 4 GPU node, which contains two NVIDIA Tesla P100 GPUs [7].

C. Weight initialisation

Weight initialisation is a crucial design choice, as it determines the starting point of the optimisation procedure. In the shallow CNN architecture proposed by Schindler et al. in [1], the authors did not specify the weight initialisation procedure they used. However, there are modern heuristics for weight initialisation that depend on the activation function used in the network. In this case, since the network uses the Leaky ReLU activation function throughout, I implemented He Gaussian initialisation, which is well-suited for networks with Leaky ReLU activations.

D. Batch size

Batch size is an important hyperparameter to consider when training deep learning models. Smaller batches give rise to longer epochs and introduce extra noise to the weight updates; however, this noise can prove beneficial if the error manifold has many deep local optima. Conversely, larger batches give rise to shorter epochs, but networks trained with large batches often struggle to generalise. Schindler et al. did not identify the batch size they used for training in [1]. Therefore, I experimented with multiple batch sizes in preliminary experiments and found that a batch size of 64 yielded the best results.

VI. Replicating Quantitative Results

Table I shows the mean accuracy my implementation achieved on the test set over five runs, along with the accuracy achieved by Schindler et al. [1] for comparison. Our results differed by approximately 3%, likely due to differences in the experimental setup and assumptions made between the two studies.

TABLE I
Accuracy achieved on the test set

Model	Epoch	Accuracy
My CNN	100	63.28
	200	64.22
Schindler et al.	100	66.56
	200	67.49

Figure 2 is a confusion matrix displaying the the performance of my implementation after 200 epochs. Notably, the matrix shows a significant difference in the per-class accuracy for different genres. For example, the network achieved a high per-class accuracy of least 80% on the blues, classical and metal genres. While conversely, it achieved less than a 50% per-class accuracy on the reggae and rock genres; in particular, it misclassified 22% of reggae songs as hip-hop.

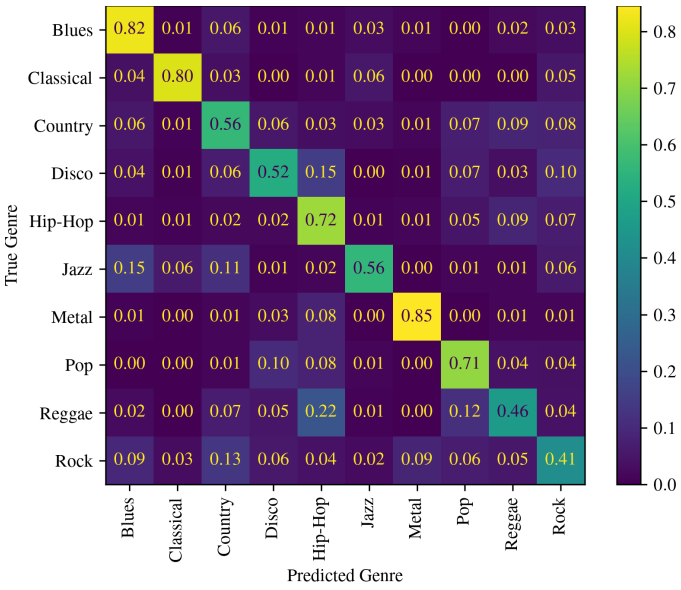


Fig. 2: Confusion matrix displaying the performance of my network on the test set after 200 epochs for a single training run. The value in a given cell represents the proportion of samples from the true genre categorised as the predicted genre.

VII. Training Curves

Overfitting occurs when a network learns the random noise in the data as if it represents the structure of the underlying model. To detect overfitting, I monitored the loss and accuracy my network of my network on both the training and test set throughout the training process. Figure 3 and Figure 4 display the accuracy and loss curves for my network, respectively.

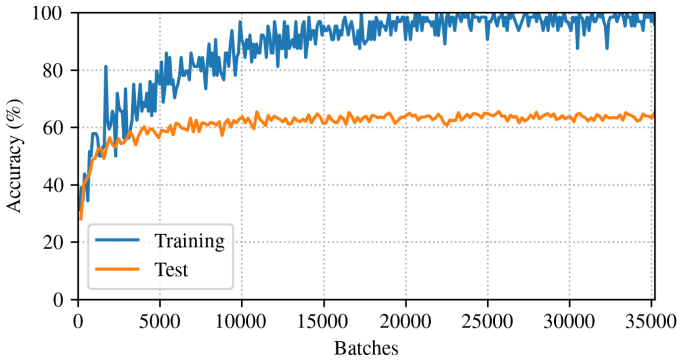


Fig. 3: Plot of accuracy data from the same training run as Figure 2. The line labelled 'training' is the accuracy achieved on the training set, calculated every 100 batches; the line labelled 'test' is the accuracy achieved on the test set, calculated every epoch.

After 200 epochs, there was a significant difference of approximately 35% in the accuracy of the model on the training and test sets—this strongly indicates that the shallow CNN architecture had overfit the training data. Even though I used L1 regularisation and dropout to mitigate overfitting, the model still memorised almost every sample in the training set, and therefore struggled to generalise to unseen data.

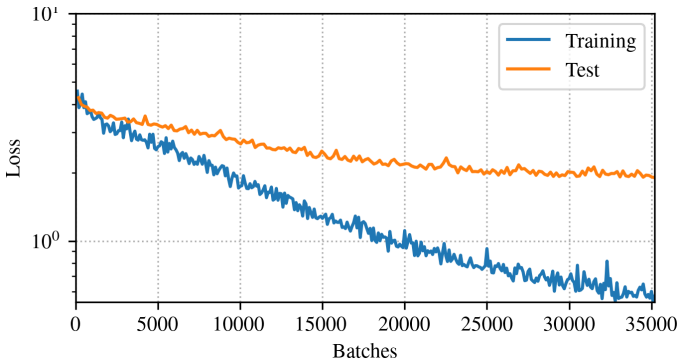
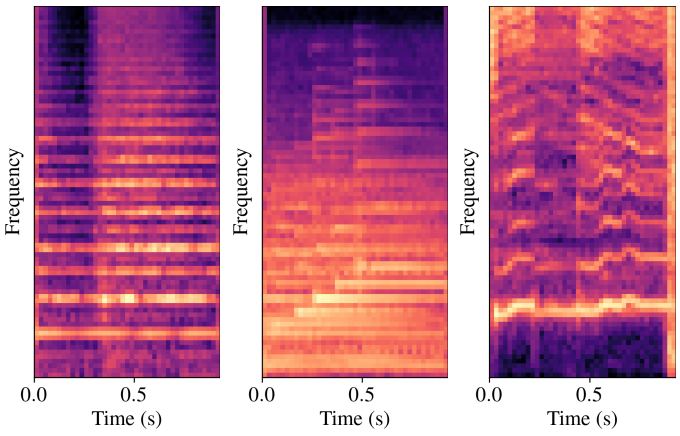


Fig. 4: Plot of loss data from the same training run as Figure 2. The line labelled 'training' is the loss achieved on the training set, calculated every 100 batches; the line labelled 'test' is the loss achieved on the test set, calculated every epoch.

VIII. Qualitative Results

Deep neural networks are frequently described as black-box models because it can be exceptionally difficult to reason about the features they extract to produce their outputs. To provide a more comprehensive understanding of the performance of the network, Figure 5 presents three log-mel spectrograms produced from samples in the GTZAN dataset to illustrate where the network performed well and where is struggled. By examining these spectrograms and listening to the corresponding audio files, I have gained an insight into the reasons for the discrepancies in per-class accuracy that the network achieved.

Figure 5a displays a correctly classified spectrogram from the classical genre. In fact, the network correctly classified all 15 spectrograms produced from the same audio file. This



(a) classical.00070.wav
Ground truth: Classical
Prediction: Classical

(b) jazz.00003.wav
Ground truth: Jazz
Prediction: Classical

(c) pop.00064.wav
Ground truth: Pop
Prediction: Disco

Fig. 5: Log-mel spectrograms produced from three of the samples in the test dataset. Each spectrogram is subcaptioned with the name of the file it was produced from, the ground-truth genre and the predicted genre.

is unsurpsising, as the music from the file is very typical of classical music, with piece being played by the string section of an orchestra. The harmonics produced by these string instruments are clearly visible in the spectrogram as extended parallel lines.

In contrast, Figure 5b displays a spectrogram from a piece of jazz music that the network misclassified as classical. It made the same error with 9 of the 15 spectrograms derived from the same piece of music. Although it is impossible to determine the exact reason for these errors, the music itself appeared to borrow elements from both the classical and jazz genres. While the piece used a jazz chord progression, there is was no syncopated beat, and it appears to contain a classical drum. Therefore, if a spectrogram was produced from a short sample that included a classical drum, it is not surprising that the network misclassified it. This highlights a potential problem with the method itself used to train and test the network, in that 0.93 second samples are often insufficient to represent a piece of music accurately.

Figure 5c shows a spectrogram produced from a piece of pop music that the network misclassified as disco. This is also unsurprising. The song has a disco beat at the beginning and first set of vocals are typical of disco music. Many pop songs have been influenced by disco, and pop artists often incorporate elements of disco into their music. This highlights a potential fundamental issue with using genre to classify music in the first place. The concept of genre itself is somewhat subjective and can vary depending on the individual and their personal taste. Many pieces of music can be classified as belonging to multiple genres, and I believe this piece fits into that category.

IX. Improvements

A. Maximum Probability and Majority Voting

As previously mentioned, each audio file in the GTZAN [2] dataset was preprocessed to produce 15 log-mel spectrograms. In order to improve the accuracy of my network, I implemented two additional classification methods that take these file dependencies into account: maximum probability classification and majority vote classification.

To classify an inputted audio file by maximum probability, the probabilities output by the final softmax layer for each spectrogram are first summed. The predicted class is then determined by the largest value amongst the summed probabilities. To classify an inputted audio file by majority vote, a class is determined for each spectrogram by the largest value output by the final layer. A majority vote is then conducted over the predicted classes for each of the segments.

These methods allow the network to consider the collective evidence from all the spectrograms for each audio file to make a more informed and accurate prediction about the genre of a given audio file. Table II displays the accuracy achieved using each of the raw, maximum probability and majority vote approaches. The relative improvement in accuracy provided by the two new approaches are consistent with the findings of Schindler et al in [1].

The increase in accuracy is due to to there being some level of independence in the predictions of samples from a given file. This independence can be attributed to the 0.93 second samples often being too short to be representative, as well as the model overfitting the training data. The majority vote and maximum probability classification methods effectively cancel out many of the individual errors to produce more accurate predictions for entire files.

Moreover, both of these strategies are frequently employed in the wider literature on music genre classification, and so make my results more comparable with similar studies.

TABLE II
Improved accuracy achieved on the test set

		Accuracy		
Model	**Epoch**	**Raw**	**Max**	**Maj**
My CNN	100	63.28	77.44	75.84
	200	64.22	77.20	76.16
My CNN + Batch Norm	100	66.57	78.88	76.24
	200	**66.78**	**79.60**	**77.68**

B. Batch Normalisation

Batch normalisation [8] is a technique frequently employed to increase the speed and stability of the training process. This is achieved via a normalisation step that fixes the means and variances of each layer's input.

I implemented a two-dimensional batch normalisation layer following each of the two convolutional layers in my network. Table II displays the accuracy of my network with the inclusion of the batch normalisation layers. Not only did batch

normalisation provide a reasonabe improvement in accuracy after 200 epochs—especially in the case of raw accuracy—but, it also sped up convergence. This is potentially due to the batch normalisation layers significantly smoothing the optimisation landscape as proposed by Santukar et al. in [9], thus inducing a more predictive and stable behaviour of the gradients which facilitates faster training. However, the precise reason cannot be determined—there is some disagreement in the wider literature on deep learning as to the theoretical basis for batch normalisation's strong empirical performance.

Batch normalisation also has a regularising effect that helps mitigate overfitting, as can be seen in Figure 6. Unlike with the network trained without batch normalisation, the network did not consistently hit 100% accuracy on the test set, and the discrepency in accuracy between the training and test set was lower at each stage. This is because each mini-batch has a slightly different mean and standard deviation, which causes the batch normalisation layers to normalise each mini-batch slightly differently. As such, there is an element of randomness, which encourages the network to be less sensitive to the random noise in its input. This overall mitigates the degree to which the network overfits.

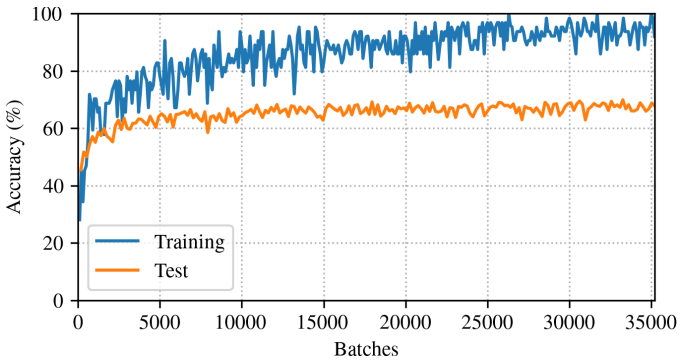


Fig. 6: Plot of accuracy data for my improved network from a single training run. Format is the same as for Figure 3.

Figure 7 displays a confusion matrix displaying the improved performance of the network following the implementation of batch normalisation. Classification was done per-file using the maximum probability method. There is an evident improvement in per-class accuracy across the genres. Notably, the network accurately classified 100% of songs from the classical genre.

X. Conclusion and Future Work

In this paper I have reimplemented the shallow CNN architecture as originally described by Schindler et al. [1]. While I was unable to reproduce the results published in the original paper, I have exposed a significant discrepency in the per-class accuracy when the the network is trained using the GTZAN dataset. Moreso, I have conducted an extensive qualitative analysis to gain an insight into potential reasons for this discrepency.

In the latter part of this paper, I have also proposed an extension to the architecture with the inclusion of two batch

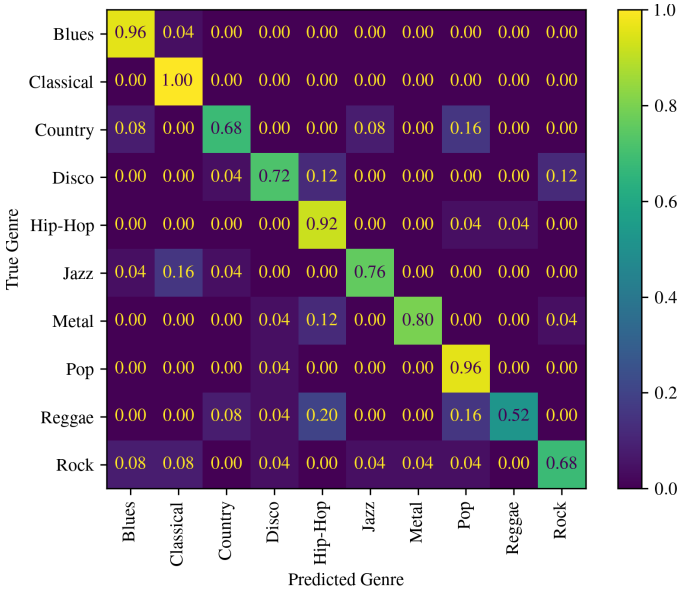


Fig. 7: Confusion matrix displaying the performance of my improved network with batch normalisation on the test set after 200 epochs for a single training run. Classification was performed using the maximum probability method. Format is the same as for Figure 2.

normalisation layers. I found that the inclusion of these layers offered an advantage over the original architecture by reducing the time required for convergence, as well as providing a modest improvement to accuracy.

Future work should focus on eliminating the stark discrepency in per-class accuracy. Further extensions should also be considered with the aim of reducing the degree to which the network overfits. Data augmentation provides a potential avenue to explore in this respect.

References

[1] A. Schindler, T. Lidy, and A. Rauber, "Comparing shallow versus deep neural network architectures for automatic music genre classification," Nov. 2016.

[2] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[3] C. Liu, L. Feng, G. Liu, H. Wang, and S. Liu, "Bottom-up broadcast neural network for music genre classification," 2019. [Online]. Available: https://arxiv.org/abs/1901.08928

[4] Y.-N. Hung, C.-H. H. Yang, P.-Y. Chen, and A. Lerch, "Low-resource music genre classification with advanced neural model reprogramming," *arXiv preprint arXiv:2211.01317*, 2022.

[5] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library."

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[7] Bluecrystal phase 4. [Online]. Available: https://www.acrc.bris.ac.uk/acrc/phase4.htm/

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[9] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" 2018. [Online]. Available: https://arxiv.org/abs/1805.11604