Image Classification Using Reduced Size DenseNet

Applied Deep Learning Course Project No 1

Maciej Manna

April 1, 2020

Abstract

Densely connected convolutional networks, abbreviated to DenseNets, constitute a reltively novel (2018) architecture of deep learning models used for image classification. Independent reviews show DenseNets to achieve at least equal, and often better accuracy than residual networks (ResNets), which are considered by many the *state–of–the–art* solution for that category of problems. These evaluation are taken under the assumption of technically unlimited resources, and models that are used have sizes and training times unconstrained by external limits. This study aims to evaluate DenseNet models of limited size and trained for limited number of epochs, and establish whether such smaller versions (with nnumber of parameters reduced about five times) can still achieve satisfying performance.

The results of chosen models of limited size were highly underwhelming. Proposed models exhibit significant bias and overfit training data, however, even if it could be reduced, obtained results seem to show that adaptation of DenseNets for limited model sizes are not a good approach in such cases. **Best achieved accuracy on test set was 0.5048.**

Introduction and Project Goals

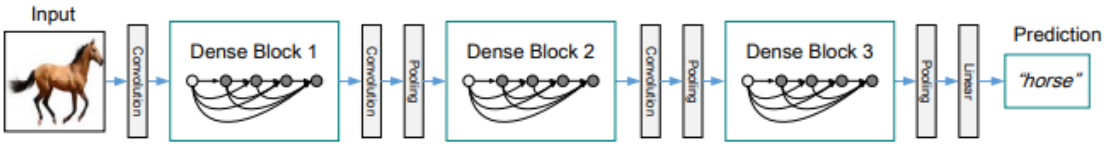
This project is an implementation of DenseNet model architecture presented in [1] using `tensorflow 2` framework, inspired by implementation in [2]. DenseNet, due to its highly connected structure, is a model with high capacity and high number of parameters. For images in `CIFAR` format ($32 \times 32 \times 3$), the proposed architecture, according to [2] ($depth = 100$, $growth\ rate = 12$) involves well over 800k parameters. Using such setup DenseNet achieves accuracy of 0.728, while even larger models ($depth = 201$) yield accuracy of 0.753 (compared to 0.740 of `resnet18`) according to [3]. The goal of this project is to evaluate how models of this architecture perform when their maximal size and time available for training are arbitrarily limited. According to project requirements, maximal number of parameters is reduced to 150k, and training time to 30 minutes.

Brief Model Overview

Model used in this project closely follows DenseNet architecture presented in [1]. As opposed to standard CNN architectures, which connect only adjacent convolutional layers (L

connections between layers), while DenseNet – in original form – connects each layer to evry other layer in a feed–forward fashion (total of $L(L+1)/2$ connections). In practice these densely–connected sets of convolutional layers are packed into several blocks (in our case 3 blocks) that are separated with transition blocks (see Figure). Structure of transition blocks (except the last one, which does not include convolution) is: BatchNormalization – ReLU – Conv 1×1 – Dropout (optional) – AvgPool. Each dense block contains some number of densely–connected convolutional blocks, that have following structure: BatchNormalization – ReLU – Conv 1×1 – Dropout (optional) – BatchNormalization – ReLU – Padding – Conv 3×3 – Dropout (optional) – Concatenate (with initial input). In addition, initial layer consists of single (3×3) convolutional layer, and before output single fully–connected layer for classification is used. All layers use $L2$–regularization with $\ell = 0.0001$.

Main two hyperparameters that affect structure of entire network is its *depth* (L), which determines the number of convolutional layers in the network. Our architecture uses 3 dense blocks, so number of convolutional blocks per dense block is calculated from given *depth* using simple formula: $(L-4)/6$. Another hyperparameter is *growth rate* (k), which determines how many filters are used in each convolutional layer in convolutional block: first one (1×1) has $4k$ filters, while second (3×3) – k.



DenseNet authors show that some of the advantages of this arcghitecture include: reduction of problems with vanishing gradient, strengthening of feature propagation, more feature reuse, and substantial reduction of number of parameters. Even though reduction of parameters is an advantage of DenseNet, its competitive models used for standard datasets usually feature one million or more parameters.

Experimental Setup

Proposed models are tested using `CIFAR100` dataset, split into training set (45k examples), validation set (5k examples), and test set (10k examples). Images from the dataset are normalized using known mean and standard deviation of `CIFAR100`.

Enforcing model parameter limit

Since recommended DenseNet models for `CIFAR`–format images incorporate around one million parameters, we must create some viable models that limit that number to 150k. Two model hyperparameters that were tweaked to bring down that number were *depth*, and *growth rate*. *Reduction* was not changed from base value of 0.5 due to its limited effect on parameter numbers. Also influence of introduction *dropout* layers was not tested.

After checking several combinations of hyperparameters that incorporated tradeoff between higher *depth*, and *growth rate*, four models with number of parameters close to 150k were chosen for further eevaluation. These models are shown in Table 1

Designation	Notebook Designation	Depth	Growth Rate	No of Parameters
DN122	`d122_gr4`	122	4	144,594
DN97	`d97_gr5`	97	5	146,866
DN56	`d56_gr8`	56	8	135,604
DN35	`d35_gr12`	35	12	151,546

Table 1: Reduced size DenseNet models chosen for further testing.

Choosing number of training epochs

To accomodate limit of 30 minutes for training time, the hard limit of training epochs was established for each model. Note, that such numbers do not represent ideal amount of time that networks of that time are usually trained (e.g. in [3] we see that at least 60 epochs with base learning rate are used, with possible additional with reduced rate). Also note, that due to such limited number of epochs no adjustments to *learning rate* were made during training and constant rate of 0.1 was used (with optimizer *momentum* of 0.9).

To calculate number of epochs each model was trained for two epochs, and time in which second one was taken into account as t_2 (as first one usually takes a little longer). Then number of epochs was calculated as:

$$epochs = \left\lfloor \frac{30 \times 60}{t_2} \right\rfloor - 1.$$

Results obtained for tested models (including actual training time measured during proper training) are shown in Table 2. Note, that time it takes for one epoch to complete grows as *depth* of the model increases.

Model	2nd Epoch Tr. Time (t_2) [s]	Training Epochs	Measured Total Training Time [s]
DN122	95	17	1628
DN97	79	21	1681
DN56	50	35	1774
DN35	42	41	1761

Table 2: Epoch number estimation and measured training times for chosen models.

Results and Analysis

Results obtained on `CIFAR100` dataset by four chosen models are presented in Table 3.

Obtained results present quite clear picture of what we were about to establish. It was obvious that the accuracies ranging from 0.72 to 0.74, that were attained by full–sized models were out of reach of our limited setups. Even considering that, accuracy scores of chosen models on test set that barely reach 0.50 are particularly underwhelming. It is quite clear that it would be a better strategy to introduce specific architecture tailored for given limitations, rather than try to adapt established DenseNet architecture.

	Traninig Set		Validation Set		Test Set	
Model	**Loss**	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**
DN122	1.91	0.60	2.56	0.47	2.5226	**0.4746**
DN97	1.88	0.61	2.54	0.48	2.5635	**0.4789**
DN56	1.81	0.63	2.74	0.46	2.7928	**0.4583**
DN35	1.81	0.65	2.59	0.50	2.5738	**0.5048**

Table 3: Loss and accuracy scores on training, validation, and test sets for chosen models..

On the other hand, proposed models themselves leave some room for improvement. Significant discrepancies between accuracies for training set, as opposed to validation and test sets, show that these models exhibit fair amount of bias and overfit training data. Such bias may be reduced by using further regularization methods wihin these models, such as inclusion of dropout layers, which were not tested due to time constraints.

In conclusion, presented models attain poor accuracy on test set around 0.5 (**best score: 0.5048**) and show limited use of DenseNet architecture when allowed model capacity and training time is highly limited. Such models show some potential of achieving accuracy slightly above 0.6, if proper regularization can be found to reduce overfitting of chosebn models, however these results are far from satisfying.

References

[1] G. Huang et al., *Densely Connected Convolutional Networks*, CVPR'17.

[2] `https://github.com/justincosentino/densenet-tf2`

[3] `https://github.com/Ecohnoch/tensorflow-cifar100`