Tiny Imagenet: Image Classification

Tatiana Matejovicova
University of Cambridge

14-12-2018

1 Introduction

The ImageNet Large Scale Visual Recognition Challenge [4] was started in 2010 to compare performance of object detection algorithms across a large variety of objects and has been used as an object recognition benchmark. Tiny ImageNet Visual Recognition Challenge [1] is an equivalent challenge of a smaller size with 200 classes. Each class has 500 training images, 50 validation and testing images, all of a size 64×64 pixels.

In this project the Tiny ImageNet database is used to design, train and test a classification deep neural network. Multiclass logistic regression and one hidden layer fully connected networks are used as a baseline and this is compared to two variations of convolutional neural networks.

2 Methods

2.1 Loss function

The loss function used in all experiments is the categorical cross-entropy.

$$\mathcal{L}(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_k^{(n)} \log p_k^{(n)}$$

Where:

- $\boldsymbol{w}$ - all model weights
- N - number of samples
- K - number of classes
- $\boldsymbol{y}^{(n)}$ - the n^{th} sample label in one-hot vector encoding i.e. $y_k^{(n)} = \mathbb{I}\left[y^{(n)} = k\right]$
- $\boldsymbol{x}^{(n)}$ - the n^{th} sample input vector
- $\boldsymbol{p}^{(n)} \in [0, 1]$ - prediction for the n^{th} sample such that $\forall n \in \{1, ..., N\}$, $\sum_{k=1}^{K} p_k^{(n)} = 1$ i.e. $\boldsymbol{p}^{(n)}$ is a probability distribution with $p_k^{(n)} = p(y^{(n)} = k|\boldsymbol{x}^{(n)}, \boldsymbol{w})$

This loss function is selected because of its suitability for multi-class classification. It measures the cross-entropy between $\boldsymbol{y}^{(n)}$ and $\boldsymbol{p}^{(n)}$. $-\log p_k^{(n)}$ can be interpreted as a surprise of the model after learning that $y^{(n)} = k$. Weights are selected to minimise the model's surprise about the training data.

$$\boldsymbol{w}* = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$$

This encourages it to match the training data.

2.2 Optimiser

Minibatch stochastic gradient descent [7] is used for all model training to decrease the computational difficulty and introduce randomness to the optimisation process. The following parameters are used.

- Batch size - 100
- Learning rate - 0.01
- Number of epochs - 40

Glorot weight initialisation [3] is done for all layers to avoid the vanishing gradient problem.

2.3 Regularisation

Overfitting occurs when the model is trained to account for features in the training set that do not generalise to other data such as noise and features not specific to a certain class. To limit this three regularisation techniques are used. When overfitting is reduced, the ability to predict on unseen data is significantly improved.

Data augmentation

Data augmentation is performed so that the model never encounters the same image twice. Augmentation performed includes rotation, translation, brightness change and a horizontal flip and it is performed randomly in a specified range for each training image in each epoch.

Note that greyscale images are used together with color images and the grey channel is extended to all three RGB channels. The effect of doing this is mitigated in the first convolutional layer.

(a) Original (b) Higher brightness (c) Lower brightness

(d) Rotation (e) Translation (f) Horizontal flip

Figure 1: Image augmentation used

Dropout

Dropout is another way to avoid overfitting in neural networks [6]. In each training step a fraction of weights in each layer is set to zero. This way a different architecture is trained in every step and neurons are prevented from co-adapting and they become multi-purpose. During evaluation an average network from all the thinned networks is used thus decreasing the magnitude of weights. Dropout reduces overfitting significantly but it makes the training procedure slower as it takes the model longer to converge. For our models, dropout of 0.2 is used for all the fully connected layers.

L_2 regularisation

L_2 regularisation penalises big weights in the model by adding the following to the loss function.

$$L_2(\boldsymbol{w}) = \lambda \sum_i w_i^2$$

$$\boldsymbol{w}* = \arg\min_{\boldsymbol{w}} \left[\mathcal{L}(\boldsymbol{w}) + L_2(\boldsymbol{w})\right]$$

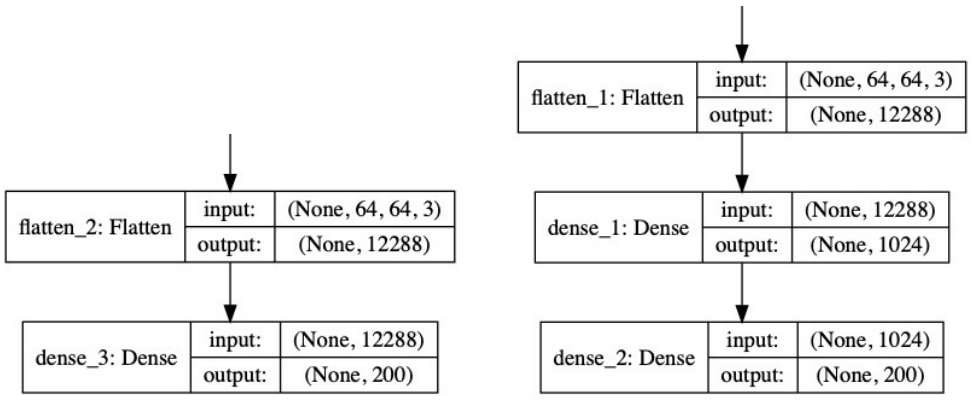
The model is discouraged from using extreme weights and thus reduces overfitting. In our model the L_2 multiplier is set to $\lambda = 0.0005$ [5].

2.4 Activation function

ReLu activation as opposed to traditional sigmoid are used to avoid the issue of vanishing gradient in all layers except for the output layer. Softmax is used in the output to layer to generate a valid probability distribution.

2.5 Baseline

Two basic models are used as a comparison baseline: a multiclass logistic regression and a fully connected network with a single hidden layer.



(a) Multiclass logistic regression

(b) Fully connected network with a single hidden layer

Figure 2: Models used as a baseline

2.6 Network architecture

The main model is a convolutional neural network. In this section its architecture is described.

Convolutional neural networks are designed to automatically extract intermediate image features such as corners, edges and blobs by having multiple hierarchical layers. This corresponds to hierarchical organisation of image object that can be decomposed to object, object parts and primitive features. CNN is typically composed of several blocks of convolution layer, activation layer and subsampling layer and the image characteristics are built in the model in the following ways.

1. Image features are translation invariant and because of that convolution weights are tied across the whole image. Each set of convolution weights corresponds to a single feature that can be localised anywhere in the image.

2. Low level features such as edges are expected to be local and so the convolution filters typically have small size such as 3×3 pixels.

3. High level features such as objects are expected to be coarser and this is accounted for by the subsampling layers such as picking the maximum value pixel in a small region.

All these image properties are built in the model so they do not have to be learnt as in fully connected networks. Therefore CNNs are more efficient in extracting deep features and require fewer parameters thanks to weight tying.

The first CNN architecture tested displayed in Figure 4 contains a single block of 2 convolutional layers followed by a second block of two fully connected layers.

1. 32 2D Convolutional filters (3,3)
   32 2D Convolutional filters (3,3)
   Max pool (2,2)

2. Flatten
   Fully connected (1024)
   Output (200)

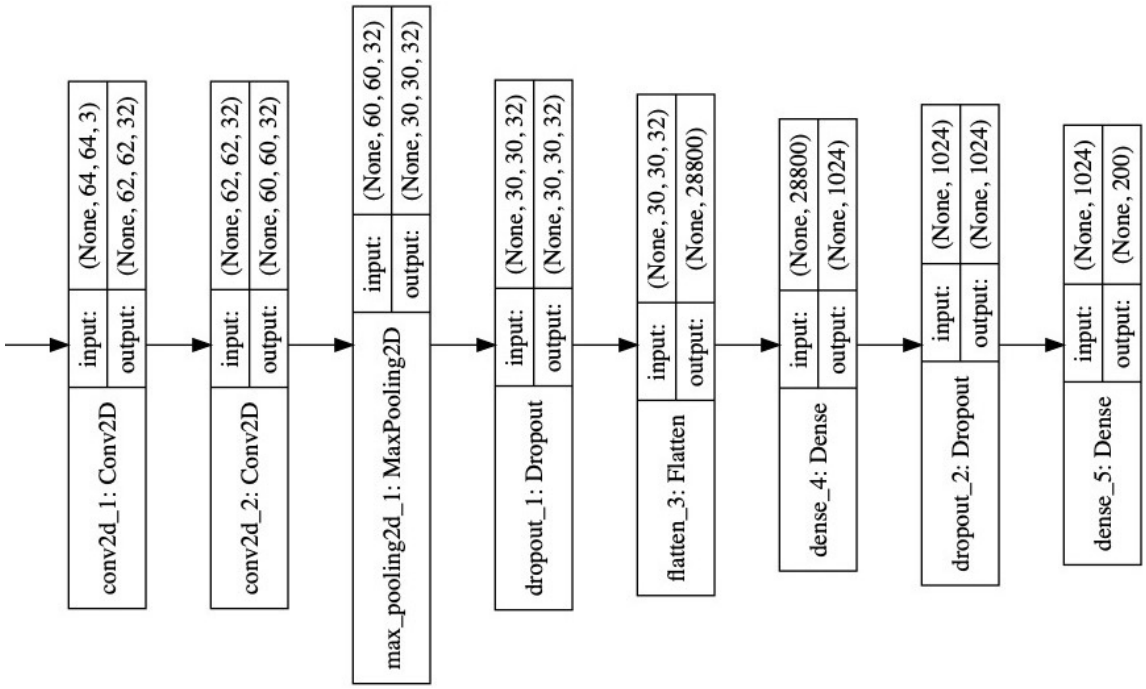


Figure 3: Simple CNN network

The more advanced CNN architecture tested in this project is inspired by the modified [2] VGG network architecture [5] that was used for the original ImageNet challenge. Its depth and number of convolution filters is reduced because of a lower number of classes that are recognised. It is displayed in Figure 2 with the following structure of four blocks.

1. 32 2D Convolutional filters (3,3)
   32 2D Convolutional filters (3,3)
   Max pool (2,2)

2. 64 2D Convolutional filters (3,3)
   64 2D Convolutional filters (3,3)
   Max pool (2,2)

3. 128 2D Convolutional filters (3,3)
   128 2D Convolutional filters (3,3)
   Max pool (2,2)

4. Flatten
   Fully connected (1024)
   Output (200)

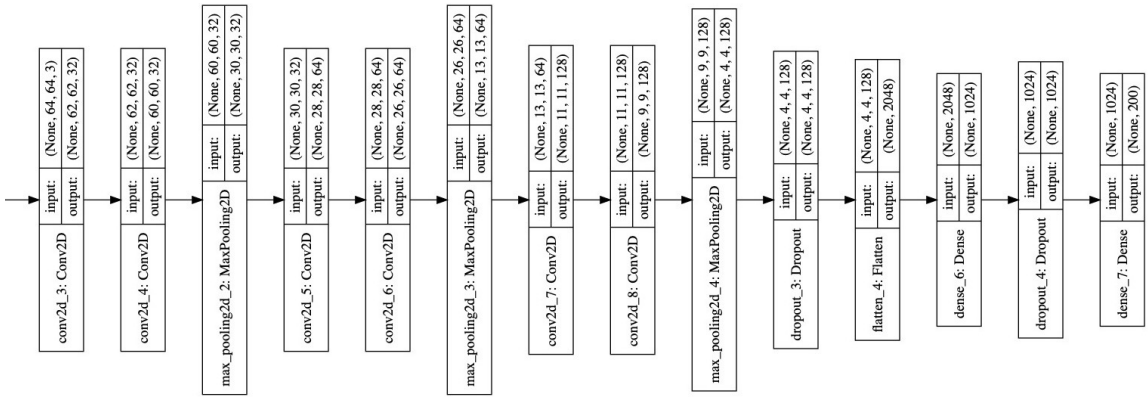


Figure 4: VGG-like CNN network

The first and last block of the two proposed CNN networks are the same but the second network has more intermediate convolution blocks. The latter the convolutional block the higher level features it is able to extract. Therefore the second network should be able to extract higher level features and perform better. The number of convolutional filters in the second network increases with each block from 32, to 64 and 128. This is because the first block detects the lowest level features and the latter blocks use their combinations to detect higher level features. The final block with two fully connected layers aggregates the features from the last max pooling layer and makes the classification decision.

2.7 Metrics

Two models with a set of weights can be compared by their loss value. On top of that two metrics are used that describe the accuracy of the model's predictions on new data.

- Top-1 accuracy - The number of correct predictions over the number of items in the test set
- Top-5 accuracy - The number of correct predictions considering the top five guesses over the number of items in the test set

2.8 Implementation

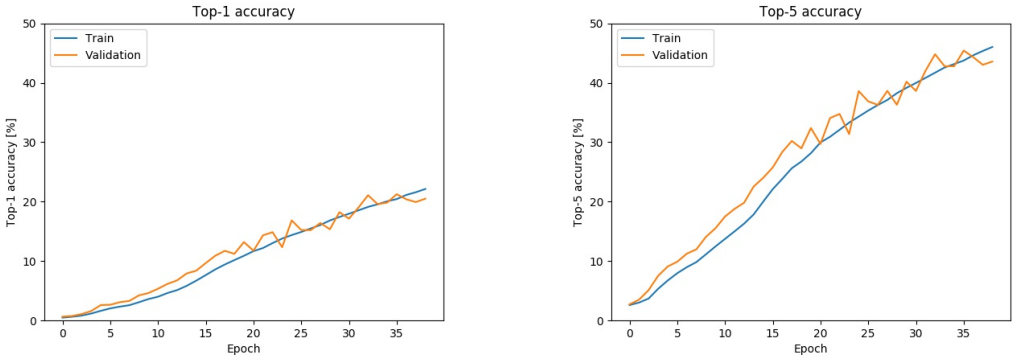
The project is implemented in a high-level machine learning language Keras with the TensorFlow backend.

3 Results

In this section quantitative and qualitative experiment results are given.

	Loss	Top-1 accuracy	Top-5 accuracy
Multiclass Logistic Regression	4.80	6.2%	17.6%
Fully connected	4.60	7.8%	22.1%
Simple CNN	4.39	21.9%	45.6%
VGG-like	3.98	24.5%	48.8%

Figure 5: Validation metrics for baseline models and CNN models



(a) Top-1 accuracy

(b) Top-5 accuracy

Figure 6: Evolution of the validation accuracy of the VGG-like model with epochs

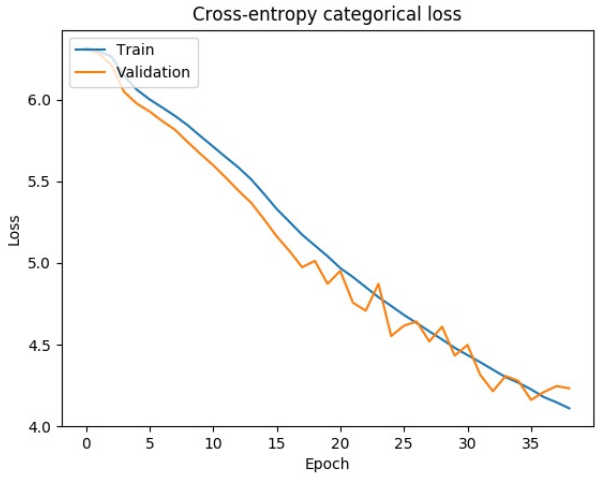


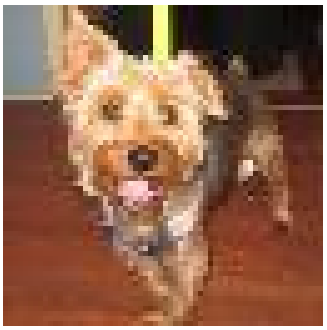
Figure 7: Evolution of the validation loss of the VGG-like mode with epochs

3.1 Classification examples

In Figure 8 classification examples are given.

(a) cauliflower, elephant, fly, bee tarantula

(b) elephant, wagon, convertible, sports car, police wagon



(c) Yorkshire terrier, kimono, vestment, basketball, bikini

Figure 8: Top five guesses for images of grasshopper, sports car and Yorkshire terrier (left to right)

4 Discussion

Firstly when observing Figure 5 it can be observed that top-1 accuracy for the baseline models logistic regression and fully connected network with one hidden layer is 6.2% and 7.8% respectively which is better than guessing equivalent to 0.5% as expected. When the baseline models are compared to the CNN architectures top-1 accuracy improves about 3-times, to 21.9% and 24.5% for the simple CNN and VGG-like network respectively.

When comparing the CNN models adding more convolutional blocks in the VGG-like architecture improved the accuracy by 2.6%. This is because the network is able to extract more higher level features such as textures and objects in the images.

When observing figures 6 and 7 the validation values are not consistently below the training values which suggests that a systematic overfitting does not occur. However after about 15^th epoch, the validation values start to fluctuate while the training values keep steadily increasing or decreasing. This is because some portion of the validation data is classified randomly. This suggests that the classifier starts to train also on the image noise rather than the underlying relationships. Overfitting could be decreased by tuning the regularisation parameters as described in subsection 2.3.

As expected the top-5 accuracy is higher than the top-1 accuracy. This is because the metric gives the model a certain leeway when classifying. For the VGG-like network the label is classified correctly in 24.5% of the images whereas it occurs in one of the top guesses in 48.8% of the images. Therefore we would expect that using this classifier about every fourth image will be classified correctly and about half of the images will be classified correctly considering the top five labels.

Finally in Figure 8 some example classifications are shown. Grasshopper is not classified correctly but in the top five guesses there are other insects fly, bee and tarantula. For the sports car, four out of five guesses are vehicles. Therefore we can observe that the object subclass can generally be guessed correctly. The third image of Yorkshire terrier is classified correctly.

It was shown that the suggested convolutional networks achieved greater accuracy than the baseline models. However the highest achieved top-1 accuracy of almost 24.5% is much lower than the highest achieved top-1 accuracy for this challenge of 73.2%. The model used could be significantly improved by adding more convolutional blocks and having more fully connected layers before the output layer. This would enable it to extract more low level and higher level features. Furthermore the gradient descent process could be improved by tuning the parameters and thus converging closer to the global optimum. The optimiser could be improved for example by using learning rate decay and momentum. Finally a pre-trained network the original data set could be utilised and fitted to this problem.

5 Conclusion

Two CNN architectures were designed to classify images from Tiny ImageNet. They were trained and compared to a baseline of two fully connected networks. The highest top-1 accuracy of 24.5% was achieved for a CNN that was based on the VGG model. The highest top-5 accuracy of 48.8% was achieved with the same model which means that it produced the correct label in the top five guesses for about half of the image samples. The performance could be improved significantly by adding more convolution blocks and increasing the number of filters. Furthermore the training procedures could be improved by tuning the optimisation parameters and using more complex techniques such as scheduled learning rate and momentum. A completely alternative approach that would be likely to yield high accuracy would be to use transfer learning.

References

[1] Tiny imagenet visual recognition challenge. https://tiny-imagenet.herokuapp.com/.

[2] Pat Coady. Vggnet and tiny imagenet. https://learningai.io/projects/2017/06/29/tiny-imagenet.html.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[7] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.