Dogs Versus Cats: A Convoluted Approach

Michael Balas, *Student, Faculty of Health Sciences*

Abstract—This paper offers a high-level overview of some of the conceptual and mathematical underpinnings of Convolutional Neural Networks (ConvNets). Techniques for improving network accuracy while preventing overfitting (such as data augmentation and regularization) are also discussed. All of these concepts are then put into practice by building an image classifier designed to distinguish between dogs and cats using a limited portion (only 8%) of the Kaggle dataset. The effects of data augmentation and network architecture on performance are examined, and results indicate that models with greater entropic capacity and extensive image modification demonstrate improved accuracy at the cost of time and computational resources.

Index Terms—Convolutional neural network, CNN, ConvNet, image classification, data augmentation.

1 Introduction

Introduced in 1989 by LeCun and colleagues, Convolutional Neural Networks (ConvNets or CNNs) are biologically-inspired variants of Multilayer Perceptrons.[1] ConvNets roughly mimic the architecture of the mammalian visual cortex, wherein they are spatially organized similarly to a retinotopic map, and convolutional and pooling layers resemble the simple and complex receptive fields of cortical cells, respectively.[2-4] This organization allows ConvNets to exploit the compositional hierarchies of natural signals. In other words, this class of feedforward neural networks is designed to process data that has a known, grid-like topology, such as that found in an image (i.e. composed of multiple two-dimensional arrays with varying pixel colour channel intensities).[4] ConvNets currently offer unprecedented performance in recognition and detection tasks, achieving near-human performance on certain problems.[4] This paper presents a high-level mathematical and conceptual overview of ConvNets and evaluates the effectiveness of this type of neural network on an image classification task for distinguishing between dogs and cats. A typical ConvNet is structured by a series of convolutional (followed by non-linearities), pooling and fully-connected layers (Fig. 1).

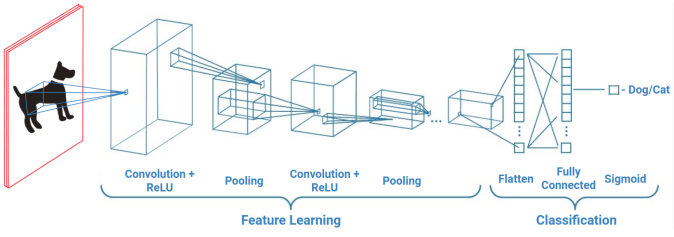

Fig. 1. The inputs (images of dogs and cats) are fed to a network of stacked convolutional and pooling layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The final output consists of a sigmoid layer which indicates whether the image is of a dog or cat.

- *M. Balas is studying under the supervision of Dr. Suzanna Becker in the Neurotechnology & Neuroplasticity Lab at McMaster University. E-mail: 1michaelbalas@gmail.com*

Manuscript received April 12, 2018

2 ConvNet Architecture

The first layer in a ConvNet is always a convolutional layer. The input to this first layer (in the context of image classification) is a $h \times w \times c$ image, where h is the height, w is the width and c is the number of colour channels (e.g. an RGB image has $c = 3$). The convolutional layer has a depth of x filters (or kernels) of size $m \times n \times d$ that extract local features of the input by moving across the image (convolving) in all possible ways (analagous to the receptive fields of cortical neurons) (Fig. 2). This process produces x outputs (or feature maps) of size $(\frac{h-m+2P}{S_h}+1) \times (\frac{w-n+2P}{S_w}+1)$, where S is the stride (distance between pixels the kernel is shifted) and P is the pad (controls the spatial size of the feature maps by padding the input image with zeroes around the border). The filtering operation performed between the neuron's weights (kernel) and input image is a discrete convolution, and can be represented mathematically for an image I and kernel K with bias b as follows:

$$conv(I,K)_{hw} = \left(b + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{d} K_{ijk} \cdot I_{h+i-1,w+j-1,k}\right)$$

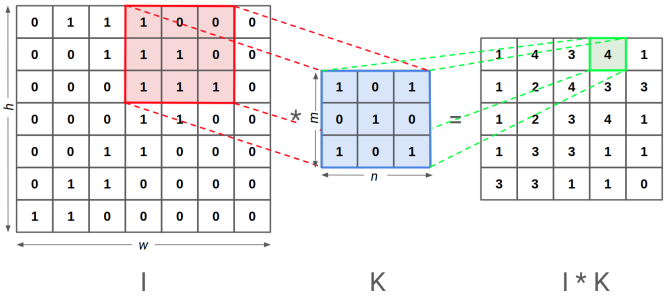

Fig. 2. Illustration of the convolution process. There is only one colour channel (c=1; d=1) for simplicity.

The kernels are determined as part of the training process via gradient descent, and unlike traditional neural networks, all units in a feature map share the same set of weights and biases (called a filter bank). Different filter banks are used for different feature maps in a layer.[4] The architecture of a convolutional layer is employed for two

reasons: (i) local groups of pixels are often highly correlated and will cooperate to form distinctive features; and (ii) features can appear anywhere within an image (position invariance).[4]

After every convolution operation, an element-wise (applied per pixel), non-linear activation function is used to signal distinct feature identification and introduce non-linearity in the ConvNet (similar to real-world data).[5] A variety of non-linearities may be used such as hyperbolic tangent and sigmoid functions, though rectified linear units (ReLUs) have been found to perform better and faster in most circumstances.[5] A ReLU is mathematically defined as:

$$Y_i^\ell = max\left(0, Y_i^{\ell-1}\right)$$

where Y_i^ℓ is the i^{th} feature map of layer ℓ. It replaces all negative pixel values in the feature map by zero, maintaining the size of inputs and outputs for each layer without affecting the receptive fields (Fig. 3).[5]

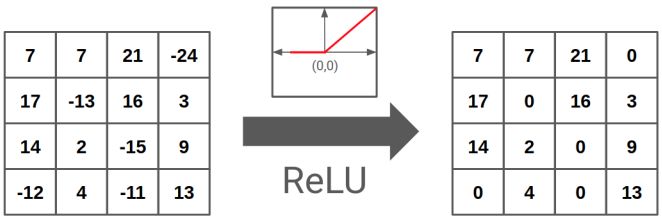

Fig. 3. Pictorial representation of ReLU functionality. All negative values are converted to zero. All others remain the same.

In contrast to the convolutional layer, the pooling (or downsampling) layer merges semantically similar features together. This layer reduces the spatial size of feature maps by consuming and aggregating small portions of the image into a single value (Fig. 4). It is used to lessen computational cost by discarding insignificant data while preserving the detected features in smaller representations, as well as minimizing overfitting by providing translational invariance.[6] There are various methods of downsampling, though max pooling has been shown to be vastly superior for capturing invariances in image-like data.[6] Max pooling operates by finding the maximum value in a local patch of pixels and discarding all other values.[7]

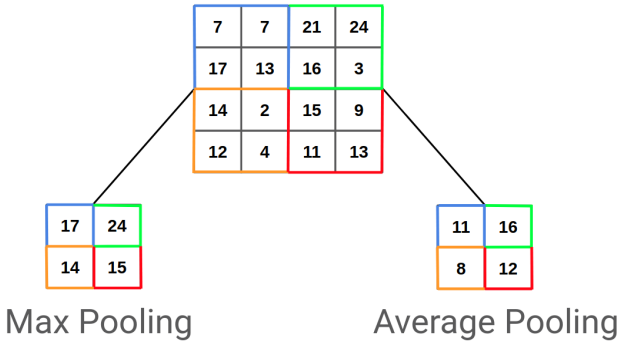

Fig. 4. Illustrative example of max pooling and average pooling.

Fully-connected layers are essentially Multilayer Perceptrons (with the exception of the input layer), wherein each neuron is connected to all elements of the previous layer, hence the name. These are often used as the final layers of a ConvNet with the purpose of computing class scores to classify input images.[5] Fully-connected layers can also be used to efficiently learn combinations of features obtained from convolutional and pooling layers.[5]

3 Methods

The ConvNets were built with Keras (version 2.0.6) using a TensorFlow backend (version 1.1.0) on an HP EliteBook Folio 9480m computer. The datasets of 25,000 cat and dog images were obtained from Kaggle, which offers a subset of the three million images from Microsoft Research's HIP (Human Interactive Proof), Asirra (Animal Species Recognition for Restriction Access).[8,9] However, only 1,000 and 400 images of each class are being used for training and validation data, respectively, to enable fast performance and illustrate the usefulness of image pre-processing.

3.1 Data Augmentation

Since only 8% of the original Kaggle dataset is being used, the ConvNet is high at risk for overfitting. One method of reducing the model's sensitivity to noise and overfitting is data augmentation, which introduces various types of random transformations in the original images to increase the size of the training dataset.[10] The augmentation techniques performed on the training dataset include rotations, translations, zooming, horizontal flipping, and shearing (Fig.5). The augmentation operations are performed iteratively in real-time to reduce memory overhead.[11]

Fig. 5. An example of what the proposed data augmentation strategy looks like. The top-left cat picture with the red border is the original image from the training dataset. All 14 other images are transformed variants of the first.

3.2 Entropic Capacity

Another method for decreasing the likelihood of overfitting is by limiting the amount of stored information, or entropic capacity, of the ConvNet.[12] Although models with large information storage may be able to leverage more features and increase classification accuracy, they are also more likely to store insignificant features that do not generalize well.[12] To reduce entropic capacity, the initial ConvNet used in the first experiments will have few parameters (i.e. a small number of layers and filters per layer). Furthermore, the

regularization technique, dropout, which was inspired from the role of sex in evolution, is used within each training case to randomly omit each unit from the network (along with its connections), with a fixed probability p (usually 0.5) independent of other units.[13] This method also reduces overfitting by reducing reliance on specific neurons and preventing complex co-adaptations on the training data.[13]

3.3 Network Architecture

Various ConvNets were built and trained to determine which configurations gave the best results. Since this is a binary classification problem, all models end with a single-unit fully-connected layer, followed by a sigmoid activation function, represented as:

$$\sigma(z) = \frac{1}{1 + e^z}$$

where z is the output from the previous layers, and $\sigma(z)$ will yield a probability value between 0 and 1. Furthermore, the binary cross-entropy loss function will be used for training the model. For a particular binary indicator vector (0 or 1), $\vec{y}$, and one-hot truth vector $\vec{\hat{y}}$, binary cross-entropy can be calculated as:

$$\mathcal{L}(\vec{y}, \vec{\hat{y}}) = -y\ log(\hat{y}) - (1 - y)\ log(1 - \hat{y})$$

The famously unpublished adaptive learning rate optimizer proposed by Geoffrey Hinton, RMSprop, was also used in all models. It works by dividing the gradient by a running average of its recent magnitudes.[14] Each ConvNet runs for 50 epochs with a batch size of 16. The metrics used to measure performance are binary accuracy and time to train the model.

4 Results

4.1 The Effects of Augmentation

In the first experiment, three models (α, β, γ) were constructed to analyze the effects of regularization and data augmentation on accuracy and overfitting. All three models have the same architecture: a stack of three convolutional layers, each with a ReLU activation function followed by max-pooling layers, ending with two fully-connected layers (the last consisting of a single unit). The first model, α, used the data as is (i.e. no augmentation) and did not perform dropout. β used dropout regularization on the final layer with p=0.5, and moderately augmented the data (i.e. random shearing and zooming with 20% intensity, as well as horizontal flipping). γ extensively modified the original images (same transformations as model β with the addition of 20% horizontal and vertical translation, and random rotation within a range of 40 degrees), and performed dropout with p=0.25 after every convolutional layer, in addition to the dropout (p=0.5) on the final layer. Indeed, the effects of overfitting gradually diminish from model α to model γ, as demonstrated by the distance between training and validation accuracy curves in Figure 6.

The validation accuracy of models α, β and γ were 70.00%, 79.25% and 75.62%, respectively. Although γ exhibited the least amount of overfitting, it only performed second-best and slightly better than α. This may be due

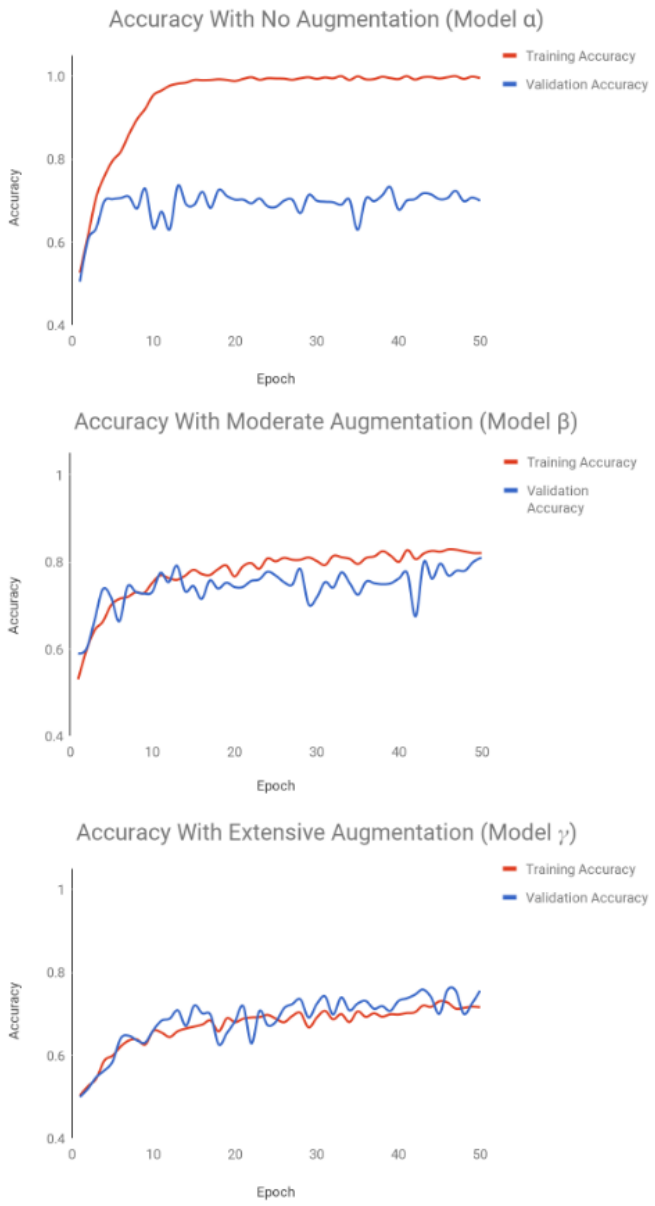

Fig. 6. Model α shows considerable difference between the training and validation accuracy, indicating that the network has attempted to memorize the training data - a classic sign of overfitting. β also shows slight overfitting, while γ shows no signs of it (in fact, validation accuracy tends to be higher than training accuracy). β exhibited the highest validation accuracy.

to the extent of data modification performed, wherein the network potentially learned features from the transformations themselves rather than the key traits of cats and dogs. However, this issue may be resolved with larger datasets and more layers in the network. All three models are trained within 13% of the same timeframe (α = 76 min, β = 76 min, and γ = 86 min).

4.2 The Effects of Layering

Using the augmentation techniques that led to the highest score in the previous simulations (Model β), the next experiment sought to determine the number of layers that were optimal for performance. Again, three models (β-1, β-2, β-3) of varying complexity and entropic capacity were

created to observe the effects on performance. Model β-1 was the smallest and simplest of the three; β-2 had the same architecture used in the first experiment (i.e. equivalent to Model β), and β-3 was the largest. The layers of all three are outlined in Figure 7.

β-1: INPUT → CONV → RELU → FC → SIG

β-2: INPUT → [CONV → RELU → POOL]*3 → FC → RELU → FC → SIG

β-3: INPUT → [[CONV → RELU]*2 → POOL]*3 → [FC → RELU]*2 → FC → SIG

Fig. 7. Abbreviations: CONV (Convolutional layer); RELU (ReLU Activation); POOL (Max Pooling Layer); FC (Fully-Connected Layer); SIG (Sigmoid Activation). Asterisks indicate repeating units.

The validation accuracy of models β-1, β-2 and β-3 were 72.88%, 79.25% and 82.13%, respectively (Fig. 8). Overfitting decreased with increasing complexity (figure not included), suggesting that models with less information storage do not generalize features well in the context of this image classification task. The time to train the models are as follows: β-1: 40 min; β-2: 76 min; β-3: 219 min. Given that β-3 is less than 3% more accurate than β-2, yet it takes almost three times as long to train, β-2 can be considered as the optimal ConvNet for this Dog/Cat classification task (especially if computing power is limited).

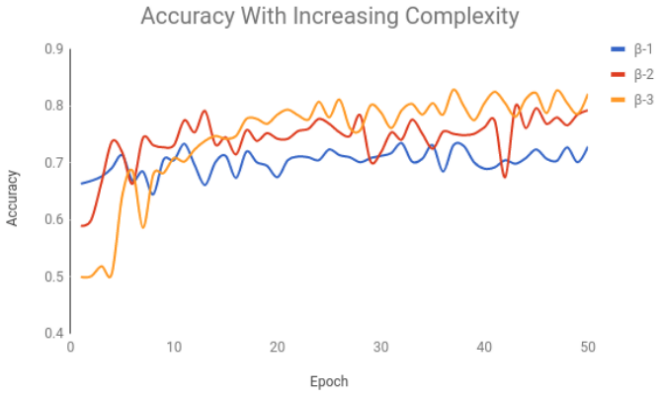

Fig. 8. Models of increasing complexity show higher validation accuracy scores and are less prone to overfitting, though computing time becomes considerably higher.

5 Conclusion

ConvNets are powerful feedforward neural networks that challenge human perception and are composed of four basic parts: convolutional layers, non-linearities (typically ReLUs), pooling layers and fully-connected layers. The limitless combinations that can be made with these layers, as well as the minimal pre-processing required to configure ConvNets, has enabled them to excel in a number of applications ranging from image recognition and video analysis to natural language processing and drug discovery.

The findings presented in this paper demonstrate that model β-2, a ConvNet of moderate complexity with three convolutional layers performs better overall compared to the other models discussed. Furthermore, data augmentation and dropout were found to have significant effects on classifier performance, resulting in as much as 10% improvements in accuracy. It is interesting to note that, using only 8% of the available data, β-2 would have already scored in the top 90 (out of 215) entrants of the Kaggle competition.[15] To achieve even better results, a more comprehensive network pre-trained on a large dataset can be leveraged (e.g. a VGG16 architecture trained on the ImageNet dataset).

Acknowledgment

The author would like to thank Dr. Suzanna Becker for her guidance and support in the undertaking of this project.

References

[1] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD. Backpropagation applied to handwritten zip code recognition. Neural computation. 1989 Dec;1(4):541-51.
[2] Ciresan DC, Meier U, Masci J, Maria Gambardella L, Schmidhuber J. Flexible, high performance convolutional neural networks for image classification. InIJCAI Proceedings-International Joint Conference on Artificial Intelligence 2011 Jul 16 (Vol. 22, No. 1, p. 1237).
[3] Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology. 1962 Jan 1;160(1):106-54.
[4] LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015 May;521(7553):436.
[5] Hijazi S, Kumar R, Rowen C. Using convolutional neural networks for image recognition. Cadence Design Systems Inc.: San Jose, CA, USA. 2015.
[6] Scherer D, Mller A, Behnke S. Evaluation of pooling operations in convolutional architectures for object recognition. InInternational conference on artificial neural networks 2010 Sep 15 (pp. 92-101). Springer, Berlin, Heidelberg.
[7] Nagi J, Ducatelle F, Di Caro GA, Cirean D, Meier U, Giusti A, Nagi F, Schmidhuber J, Gambardella LM. Max-pooling convolutional neural networks for vision-based hand gesture recognition. InSignal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on 2011 Nov 16 (pp. 342-347). IEEE.
[8] Dogs vs. Cats [Internet]. Kaggle. 2013. Available from: https://www.kaggle.com/c/dogs-vs-cats/data
[9] Elson J, Douceur JJ, Howell J, Saul J. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization.
[10] Tabik S, Peralta D, Herrera-Poyatos A, Herrera F. A snapshot of image pre-processing for convolutional neural networks: case study of MNIST. Int J Comput Intell Syst. 2017 Jan 1;10:555-68.
[11] Image Preprocessing [Internet]. Keras Documentation. Available from: https://keras.io/preprocessing/image/imagedatagenerator-class
[12] Chollet F. Building powerful image classification models using very little data. Retrieved December. 2016 Jun;13:2016.
[13] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research. 2014 Jan 1;15(1):1929-58.
[14] Hinton G, Srivastava N, Swersky K. Neural Networks for Machine Learning-Lecture 6a-Overview of mini-batch gradient descent.
[15] Dogs vs. Cats Leaderboard [Internet]. Kaggle. Available from: https://www.kaggle.com/c/dogs-vs-cats/leaderboard