# Machine Learning Practical - Coursework 3

Luca Scimeca
UNN: s1344727

February 25, 2017

# Contents

# 1 Introduction

This report describes the experiments run for *Coursework 3* of the course Machine Learning Practical at the University of Edinburgh. The coursework, first of a set of two experiments, involves the training and evaluation of Neural-Networks (NN) to perform object recognition in the CIFAR-10 and CIFAR-100 datasets. In these experiments we establish a baseline network performance to improve on in the second set of experiments.
We focus on two different aspects to improve performance on a baseline feed forward Neural Network: in the first we explore and compare performance differences when training the network with various learning rules; in the second we compare performance improvements when using Auto-encoders to pre-train the Neural Network and when using dropout to *turn off* units at random during training.

# 2 Methods

The networks in these set of experiments are constructed and run on *TensorFlow* version 0.12.
The experiments are run on the CIFAR-100 and CIFAR-10 datasets containing 60000 3x32x32 pictures of various objects. All images in CIFAR-100 are classified into one of 100 fine grained categories (10 for CIFAR-10) and 20 coarse *super-classes* incorporating multiple fine grained classes.
For the purpose of these sets of experiments we train the models on a training set of 40000 images, splitting the remaining 20000 equally into a validation and test set. At training time, all models are fit with mini-batch Gradient Descent, with a batch size of 50, and are run for 100 epochs over the training data. Unless otherwise specified, the weights are initialized with a GlorotUniform initialization and the error is computed as the multi-class cross entropy error (*nn.softmax_cross_entropy_with_logits* in *TensorFlow*).
To avoid repetition, the following sections explain the experiments when applied to the CIFAR-100 datasets. However, the same experiments were run on both CIFAR-10 and CIFAR-100; for each experiment, the equivalent results on CIFAR-10 are reported in Appendix II.

## 2.1 Learning rules

The first set of experiments focuses on comparing the performance of a baseline feed-forward network when trained to classify objects in the CIFAR-100 dataset, with different learning rules. The network designed is a simple network with a input layer of 3027 (each unit consisting of a pixel in a CIFAR image), two fully connected hidden layers of respectively 400 and 200 hidden units and a final output layer of 100 units. The input and hidden layers in the network perform a non-linear *ReLu* transformation to their units, while the output applies a *SoftMax* to its affine layer, effectively performing a 1-of-100 classification of the inputs.
To be able to compare Network performance when trained with different learning rules, a hyper-parameter search needs be done in turn for each of the learning rules, and only the best performing parameters used for the final runs. Given the computing load of searching over each hyper-parameter space in turn for each learning rule, two further sets are declared, i.e. a *mini_train_set* and *mini_valid_set*. In the following experiments, for each of the hyper-parameter searches the models are trained on the *mini_train_set*, containing a subset of 10000 images, and validated on the *mini_valid_set* containing a subset of 3000 images from the CIFAR datasets. Once the hyper-parameters for the learning rules are chosen, each model is finally trained and its performance compared on the original training and validation sets.

### 2.1.1 Simple Gradient Descent

The first learning rule to be explored is *Simple Gradient Descent*. The Gradient Descent learning rule is applied to the network designed by specifying the *TensorFlow* optimizer *GradientDescentOptimizer*. In Gradient Descent, the gradient for each weight $w_i$ is computed as:

$$\Delta w_i(t) = -\eta \frac{\partial E}{\partial w_i(t)} \tag{1}$$

the learning rate $\eta$ controls how big of a step to take in the direction towards the minimum of the error in weight space [1]. To find the best $\eta$ settings the network is trained 10 times over the data, each time performing 100 epochs with a different learning rate hyper-parameter from 0.0001 to 0.8. Figure 1 shows the minimum validation error for each fit with a different $\eta$ value. The hyper-parameter value which minimized the validation error on the *mini_valid_set* for CIFAR-100 was found to be 0.005 (see Figure 1 - Figure 9, Appendix II for CIFAR-10).

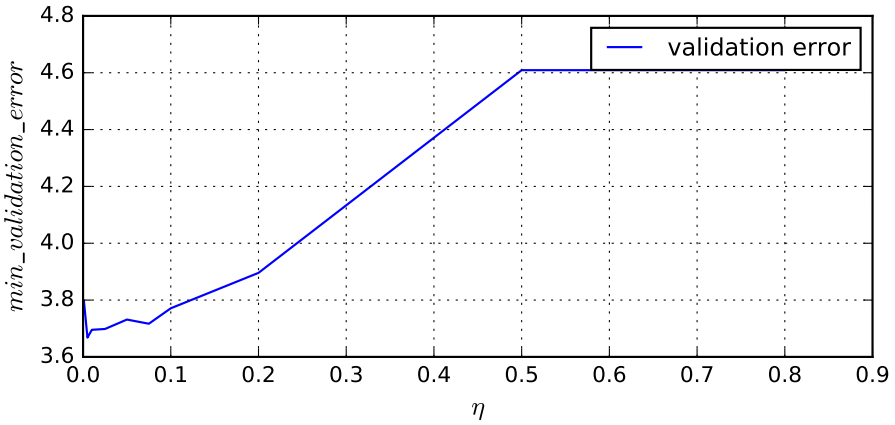


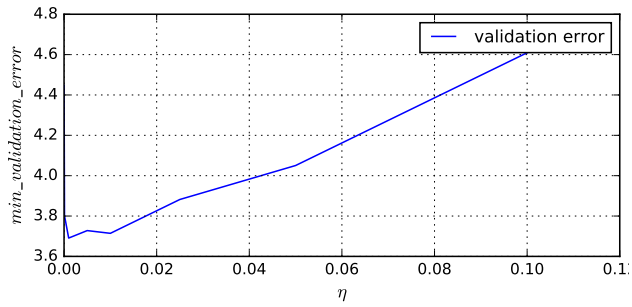

Figure 1: The Figure shows the change in *minimum* validation error of the devised NN when trained with simple Gradient Descent on the CIFAR-100 *mini_train_set*, with varying values of $\eta$. The validation error is computed on the *mini_valid_set* validation set. The value of the hyper-parameter which minimizes the validation error is $\eta = 0.005$.

### 2.1.2 Momentum Learning Rule

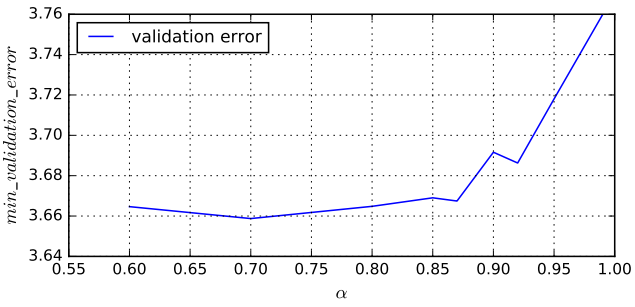
The second learning rule to compare is Gradient Descent with momentum. The momentum learning rule is applied to the network through the use of the *MomentumOptimizer TensorFlow* optimizer. The gradient for the momentum rule is computed as:

$$\Delta w_i(t) = -\eta \frac{\partial E}{\partial w_i(t)} + \alpha \Delta w_i(t-1) \tag{2}$$

where the moment coefficient $\alpha$ is here another hyper-parameter which needs setting [1]. A good starting point for the $\alpha$ value is 0.9 (MLP Lecture 4, Slide 6), thus this is fixed while an hyper-parameter search over the learning rate is again performed. Figure 2a shows the minimum validation error achieved by each setting of $\eta$, yielding the best performance for $\eta = 0.001$.
Once a good value for the learning rate is found, this can be fixed, and a search can be done around the known good $\alpha$ value, achieving a best overall performance for $\alpha = 0.7$ (see Figure 2b - Figure 10, Appendix II for CIFAR-10)).




(a) Eta search.




(b) Alpha search.

Figure 2:
Figure (a) shows the change in *minimum* validation error when the NN devised is trained with a Momentum learning rule, fixing the $\alpha$ hyper-parameter to 0.9 (a known a-priory good value) and searching over the $\eta$ space. The value which minimizes the validation error on the *mini_valid_set* set for CIFAR-100 is $\eta = 0.001$
Figure (b) shows the change in validation error when changing the $\alpha$ hyper-parameter while fixing $\eta = 0.001$. The best performance on the *mini_valid_set* is given for $\alpha = 0.7$.

### 2.1.3 RMSProp

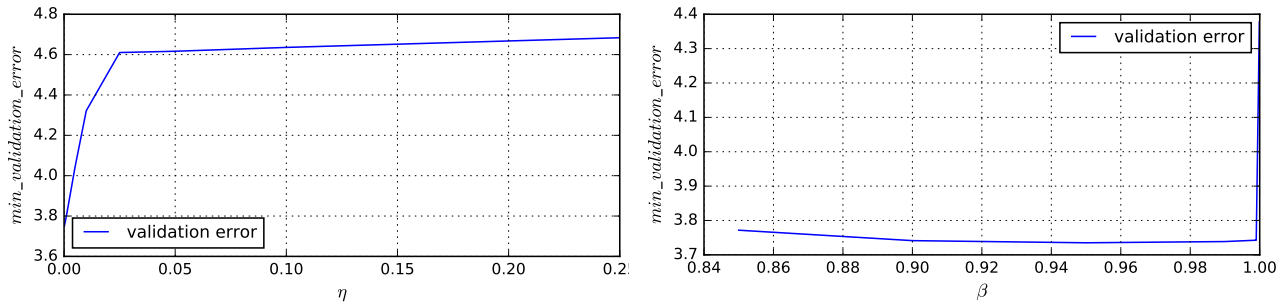
Moving on from simple Gradient Descent and Momentum rules, it is worth exploring performance for more interesting learning rate scheduling techniques, we thus shift our attention to adaptive learning rules. The first of such rules is RMSProp, where the weight gradient is computed as:

$$\Delta w_i(t) = \frac{-\eta}{\sqrt{S_i(t)} + \epsilon} \frac{\partial E}{\partial w_i(t)} \tag{3}$$

where

$$S_i(t) = \beta S_i(t-1) + (1-\beta) \frac{\partial E}{\partial w_i(t)}^2 \tag{4}$$

Here $D_i(t)$ is the gradient of the error function with respect to $w_i$ at time t, $S_i(t)$ a weighted average of the square sum of the gradients up to t, and $\epsilon$ a value to prevent division by 0 [1], set to $1e-10$ throughout the rest of the experiments.
The best hyper-parameter settings in RMSProp are found similarly to other experiments. Initially $\beta$ is set to 0.9, a good initial value for the hyper-parameter (MLP Lecture 4, Slide 9), and a search on the $\eta$ space is performed, yielding the best performance for $\eta = 0.0001$ (see Figure 3a). Afterwards, $\eta$ is fixed at 0.0001 and the $\beta$ hyper-parameter searched around 0.9 to find the setting which would minimize validation error, i.e. $\beta = 0.95$ (see Figure 3b - Figure 11, Appendix II for CIFAR-10).




(a) Eta search.

(b) Beta search.

Figure 3:
Figure (a) shows the change in *minimum* validation error when the NN devised is trained with the RMSProp learning rule, fixing the $\beta$ hyper-parameter to 0.9 (a known a-priory good value) and searching over the $\eta$ space. The value which minimizes the validation error on the *mini_valid_set* set for CIFAR-100 is $\eta = 0.0001$
Figure (b) shows the change in validation error when changing the $\beta$ hyper-parameter while fixing $\eta = 0.0001$. The best performance on the *mini_valid_set* is given for $\beta = 0.95$.

### 2.1.4 Adam

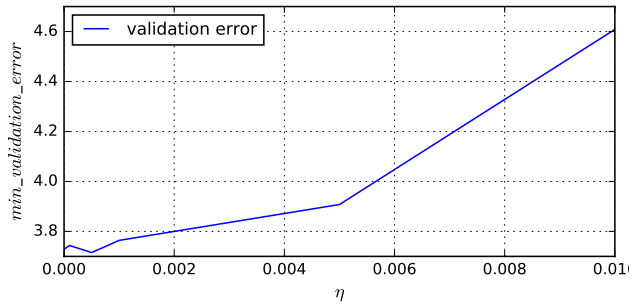
The last learning rate rule, Adam, is applied through the *TensorFlow AdamOptimizer*. Adam can be considered an extension of RMSProp, where a *momentum* biased parameter is substituted to the previous derivative of the error with respect to the weight, thus:

$$S_i(t) = \beta_1 S_i(t-1) + (1-\beta_1)\frac{\partial E}{\partial w_i(t)}^2 \tag{5}$$

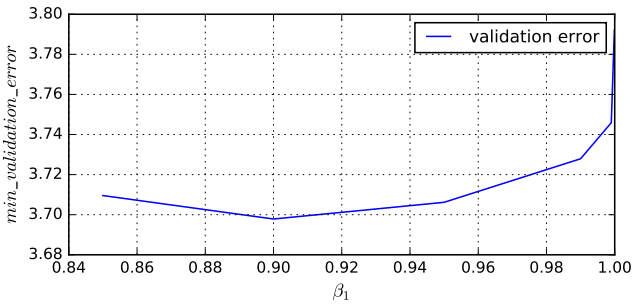
$$M_i(t) = \beta_2 M_i(t-1) + (1-\beta_2)\frac{\partial E}{\partial w_i(t)} \tag{6}$$

$$\Delta w_i(t) = \frac{-\eta}{\sqrt{S_i(t)} + \epsilon} M_i(t) \tag{7}$$

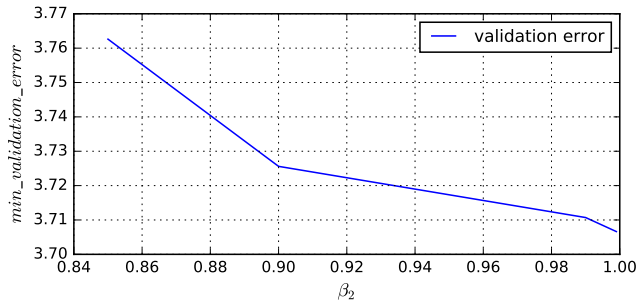
The hyper-parameters are now $\beta_1$, $\beta_2$ and $\eta$, which need setting. Here, $\epsilon$ is again set to $1e-10$ to prevent division by 0 [1]. To find the best set of values for the hyper-parameters, we perform a search for each parameter in turn. Initially we set $\beta_1 = 0.9$ and $\beta_2 = 0.999$, two known good values for the hyper-parameters (MLP Lecture 4, Slide 10), and perform a search over the $\eta$ value, finding the best performance for $\eta = 0.0005$. After finding $\eta$ we search over $\beta_1$ setting $\eta = 0.0005$ and $\beta_2 = 0.999$ and find the value minimizing the validation error at $\beta 1 = 0.9$. Finally we search over the $\beta_2$ space, setting the other hyper-parameters to the found good values and observing a minimum in the validation error for $\beta_2 = 0.9999$. Figure 4 shows the change in minimum validation error when searching over each of the hyper-parameter spaces in turn (see Figure 12, Appendix II for CIFAR-10).




(a) Eta search.




(b) Beta1 search.




(c) Beta2 search.

Figure 4:
Figure (a) shows the change in *minimum* validation error when the NN devised is trained with Adam learning rule, fixing $\beta_1 = 0.9$ and $\beta_2 = 0.999$ (two known a-priory good values) and searching over the $\eta$ space. The value which minimizes the validation error on the *mini_valid_set* set for CIFAR-100 is $\eta = 0.0005$
Figure (b) shows the change in validation error when changing the $\beta_1$ hyper-parameter while fixing $\eta = 0.0005$ and $\beta$_=0.999. The best performance on the *mini_valid_set* for CIFAR-100 is given for $\beta = 0.9$.
Figure (c) shows the change in validation error when fixing $\eta = 0.0005$ and $\beta_1 = 0.9$, and searching over the $\beta_2$ hyper-parameter space. The minimum validation error on the *mini_valid_set* for CIFAR-100 validation set is reached for $\beta_2 = 0.9999$.

## 2.2 Auto-encoders

Auto-encoders are Neural Networks designed to learn a *representation* of the data, by training to reproduce in output the input given to them. As the inputs in feed-forward network are composed of single sequential pixels, there is not much space for the learning contextual information. The aim, here, it to learn a *useful* representation of the images in the hidden layers of the network, and use that representation to categorize each picture into 1-of-100 categories. We investigate if by utilizing an Auto-encoder to pre-train the first and second hidden layer of the NN designed in the previous section, it is possible to achieve an improvement in classification performance.
To test this hypothesis we create three NNs. The First neural network possesses an input layer of 3027 units, fully connected to one hidden layer of 400 units, which after applying a ReLu transformation to its outputs feeds into an affine layer, remapping the 400 units into a 3027 dimensional output. The targets for the network are the inputs themselves, and the error to minimize is the

mean square sum of the differences, i.e.

$$\bar{E} = \frac{1}{D} \sum_{i=0}^{D-1} (y_i - t_i)^2 \tag{8}$$

where D is the dimensionality of the data (here $D = 3027$), for each element in the batch. The network is initialized with a GlorotUniform initialization and trained over 70 epochs after which its weights state is saved. The second network comprises an input and output layer identical to the previous network, and 2 hidden layers. The first hidden layer is a copy of the hidden layer in the previous network and is thus initialized to its state after training; the second is composed of 100 units, and like the first, it applies a ReLu transformation to its outputs. After initializing the second hidden layer with a GlorotUniform initialization, the network is trained to recognize its inputs over 70 epochs through the data. Ultimately the trained hidden units are used in the final network, composed of an input layer and two hidden layers like previously described, and initialized to the state of the hidden layers in the second network, after training. The second hidden layer in the network is connected to an output layer of 100 units, classifying the images into 1-of-100 classes. The targets for the network are thus changed from the inputs to the target classes in the data. Figure 5 gives an overview of the process.

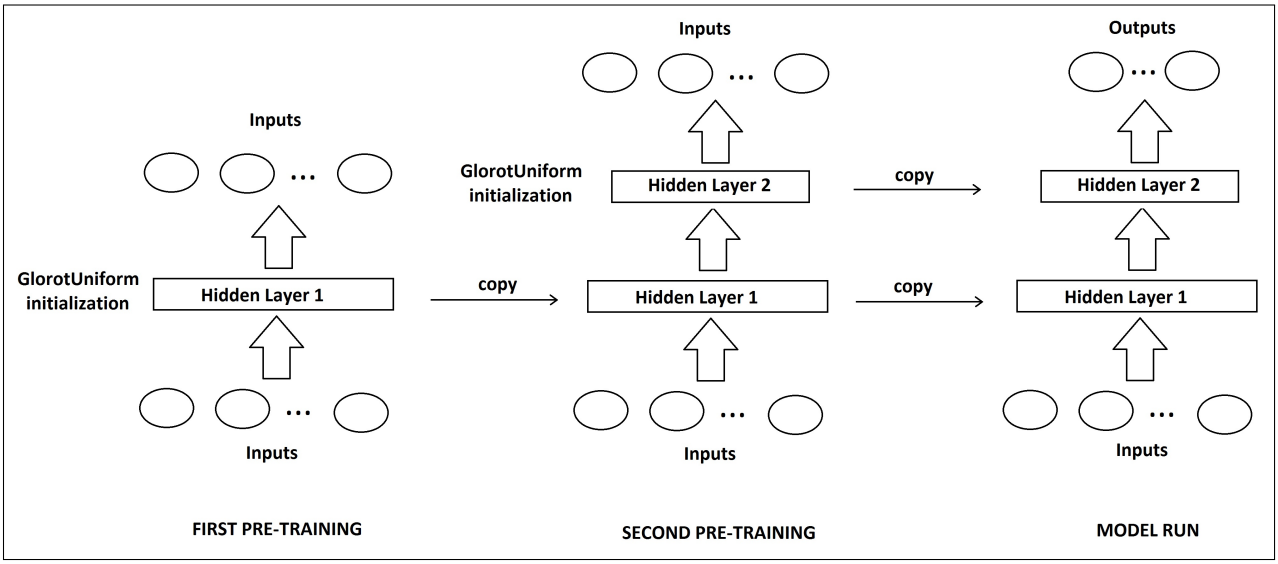



Figure 5: The Figure shows the training process used for pre-training the simple Gradient Descent model described in section 2.2. The hidden layers are pre-trained incrementally in a Neural Network to reproduce the images given in inputs. At each pre-training stage, the hidden layer previously trained is copied in the new network to be re-trained together with the additional hidden layer, initialized with a GlorotUniform initialization. Finally all hidden layers are copied in the previously designed network to be train for object classification on the CIFAR-100 and CIFAR-10 datasets.

## 2.3 Dropout

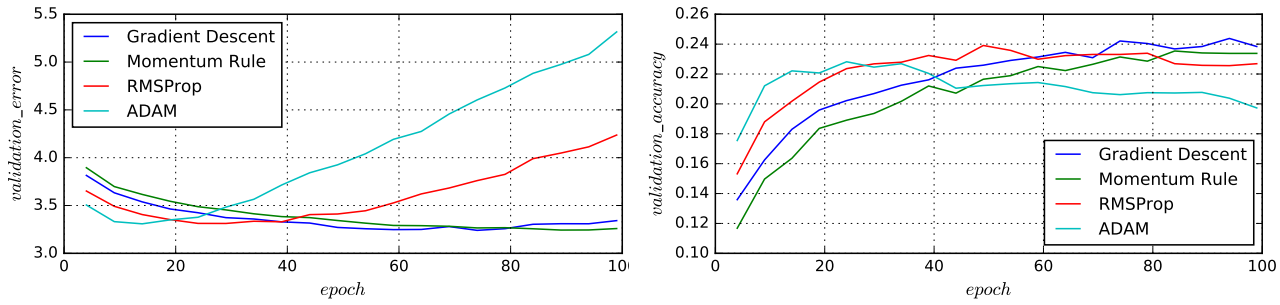
When training NN to do object classification, we wish to extract those features in the data which allow us to clearly distinguish one category from another. However, when minimizing the error against the training set, we might be learning features which may work for that particular set, but

will not generalize well to future data. For example, if the majority of the images categorized as *truck* happened to have a red truck as object of the image, then perhaps classification performance on the set is inherently improved by looking for red objects and classifying images containing those as belonging to the *truck* category. Although comparing models on a validation set helps reducing some of this problems, the possibility of poor generalization performance is still high. To *force* the network not to rely on any one particular feature as in the example, but instead to extract more *useful* and discriminatory features from the data we use dropout. With dropout, at training time we keep a hidden unit with a *keep_probability*, and drop it otherwise (i.e. the unit outputs 0). Dropping units allows for less hidden units to tune themselves to occasional discriminatory features and forces the network to learn more interesting characteristics of the data, essentially having a regularizer effect, and intrinsically preventing over-fitting. The previously described network is here augmented; the two hidden units are wrapped with an *nn.dropout* layer, while the remaining features are otherwise untouched.

# 3 Results

## 3.1 Learning rules

The simple feed forward network was designed to perform object recognition on the CIFAR dataset. To compare the performance of the network when trained with different learning rules, the hyper-parameters for each rule were searched in turn, and the settings which minimized the validation error on a subset of the validation dataset were chosen.
When trained with a simple Gradient Descent learning rule, the network achieved a validation error of $e = 3.667828$ for $\eta = 0.005$. With a Momentum learning rule, the validation error reached $e = 3.658750$ for $\eta = 0.001$ and $\alpha = 0.7$. The adaptive learning rules were overall slightly less performing, RMSProp attained an error $e = 3.735100$ for $\eta = 0.0001$ and $\beta = 0.95$, slightly more than Adam reaching a minimum error of $e = 3.706621$ for $\eta = 0.0005$, $\beta_1 = 0.9$ and $beta_2 = 0.9999$. Table 1 (Appendix I) gives an overview of the results when performing the hyper-parameter search for each learning rule (see Table 3, Appendix II for CIFAR-10).
After fixing the hyper-parameters to the optimal values previously found, the model is trained, with each learning rule in turn, on the full training set of 40000 images and validated on the 10000 images reserved for validation. Figure 6 shows the performance of each model on the validation over 100 epochs (Figure 13, Appendix II for CIFAR-10). As the figure shows, simple Gradient Descent, when the learning rate hyper-parameter $\eta$ is properly set, gets us far. The model trained with Gradient Descent reaches an accuracy of 24.51%, indeed achieving the best performance over all previous models. Training the model with momentum allows for the classification error on the validation set to reach its minimum over all other models, although classifying somewhat less accurately than the model trained with Gradient Descent. The adaptive learning rules result overall less performing. The model trained with RMSProp reaches a similar performance to that of Gradient Descent, but results in a considerable increase in training time ($\approx 110sec$). The model trained with Adam results considerably less performing than all other models on CIFAR-100. The learning rule however, reaches its top performance after $\approx 10$ epochs, and could therefor be more time efficient, given its potential for early stopping.

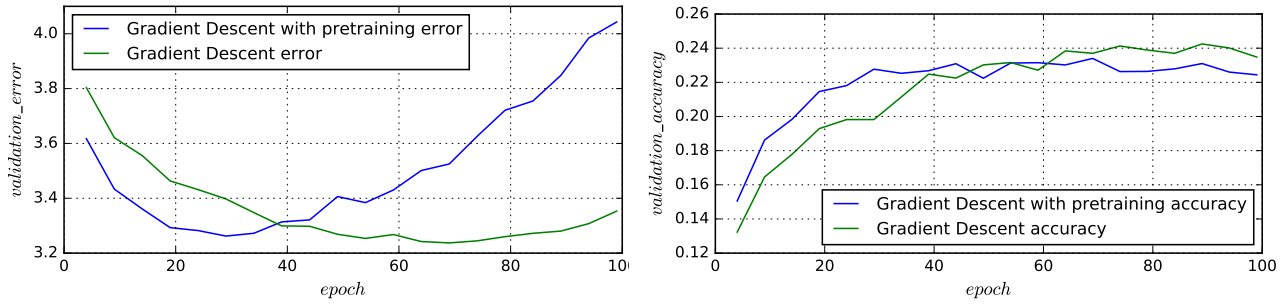



(a) Error (b) Accuracy

Figure 6:
Figure (a) and (b) show the change in, respectively, validation error and accuracy when the NN devised in Section 2.1 is trained with Gradient Descent, Momentum, RMSProp and Adam optimizer over 100 epochs on the CIFAR-100 dataset. Each model is trained with the optimal hyper-parameter settings previously found through a systematic hyper-parameter search.

### 3.2 Auto-encoders

After pre-training the hidden layers of the network designed to reconstruct images, this is run over 100 epochs through the training dataset and its performance on the validation data is recorded and compared to a simple Gradient Descent performance. As clear from Figure 7 (Figure 14, Appendix II for CIFAR-10), the pre-trained model reaches its best performance much sooner ($\approx 30 epochs$) than when randomly initialized with a GlorotUniform initialization. The pre-training allows the hidden layers of the network to start from a *better* position in weight space than when randomly initialized. Although the error appears close to that previously achieved by randomly initializing the network, the final classification accuracy of the pre-trained model does not out-perform the prior models (see Appendix I, Table 2), and is therefor discarded as baseline for future experiments.

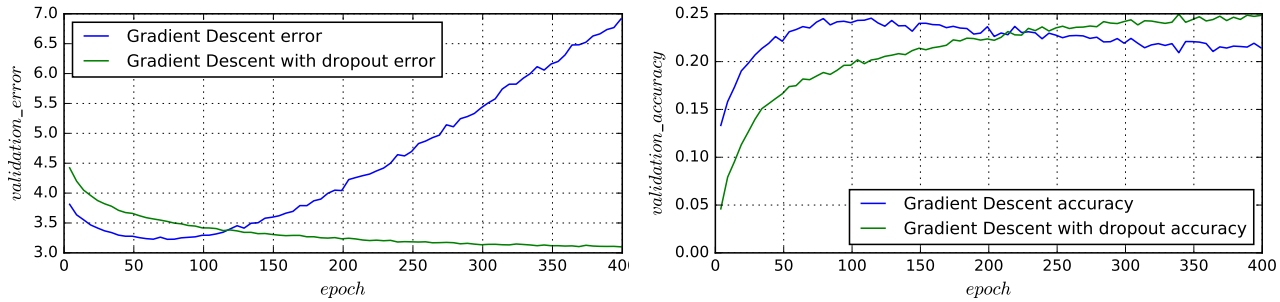



(a) Error (b) Accuracy

Figure 7: The Figure shows the change in validation error and accuracy of the model trained with Gradient Descent on the CIFAR-100 dataset, after its hidden unites were systematically pre-trained to recognize the images in input. The network reaches its best performance sooner than when randomly initialized but does not improve on the baseline top performance of simple Gradient Descent on a network initialized with a GlorotUniform initialization.

### 3.3 Dropout

After wrapping the hidden layers of the network previously designed with a *nn.dropout* layer, we train the model on the training data and validate on the validation set of 10000 images. At training time, the *keep_probability* is set to 0.5, thus giving each hidden unit an equal probability of being kept or dropped on every iteration. At validation time, the probability is rescaled back to 1.
Figure 8 shows the performance of the model trained with a dropout layer to the simple (yet so far best performing) Gradient Descent model (Figure 15, Appendix II for CIFAR-10). As clear from the Figure, dropout reaches a better performance than the initial simple model both in terms of validation accuracy and error. However, the performance increase comes at a price. The model, in fact, needs to be trained over more than twice as many epochs as other models to reach similar performance, a drawback induced by each hidden unit being deactivated on $\approx 50\%$ of the iterations through the data, which in turn requires more time to be trained.




(a) Error

(b) Accuracy

Figure 8: The Figure shows the change in Error (a) and Accuracy (b), when training the Neural Network with dropout over 400 epochs through the CIFAR-100 training set, and comparing to a simple Gradient Descent model. Dropout allows the network to reach a better validation accuracy and error, and shows a better asymptotic behavior over epochs. The introduction of dropout, however, introduces training delays, as $\approx 50\%$ of all the units in the hidden layers are inactive at any given point (thus lengthening gradient training for said hidden units of approximately twice the time).

## 4 Discussion

To find a baseline model to compare to in the second set of experiments, we try to improve performance on the CIFAR-100 and CIFAR-10 datasets when implementing simple feed-forward Neural Networks. The simple model initially designed is trained with various learning rule and their performance on the data is compared. Although different rules benefit training in a way or another, as sheer performance is the concern of the first set of experiments, the model trained with simple Gradient Descent is shown to classify objects best in the dataset. It is important, however, to note the circumstances which allowed Gradient descent to perform well, most importantly the GlorotUniform initialization and the right choice of learning rate (Adaptive Learning rules would be more robust against different initializations - see [2]).
To improve on the simple Gradient Descent model we explore two further possibilities. In one, we attempt to improve performance through pre-training the hidden layers with autoencoders to try and extract useful representation of the data in the hidden layers of the designed network; in the other, we augment the network with dropout layers, so to avoid any one hidden unit to tune itself to

noise in the data. Although reaching its top performance sooner, pre-training does not out-perform the baseline model. The network starts in a better position in weight space, but ultimately reaches similar performance to that of the simple Gradient Descent model. Augmenting the model with dropout layer improves both on validation accuracy and classification error out-performing the simple Gradient Descent based model (see Table 2, Appendix I) and reaching a classification accuracy of 24.96% on the validation set for CIFAR-100.
Although the report was centered on CIFAR-100, the algorithms were additionally run on CIFAR-10 and a similar performance throughout the experiments was observed on the different dataset. As previously, autoencoders seemed not to improve classification performance, and dropout appeared to be the most performing architecture, reaching a classification accuracy of 54.32% on the CIFAR-10 validation set. The drop out architecture, and resulting performances will therefor used as baseline for the second set of experiments.

## References


[1] L. Scimeca. Machine learning practical - coursework 1. 11 2016.

[2] L. Scimeca. Machine learning practical - coursework 2. 12 2016.

[3] S. Renals. Machine learning practical lecture 4.
`http://www.inf.ed.ac.uk/teaching/courses/mlp/2016/mlp04-learn.pdf`, October 2016.

## Appendix I: CIFAR 100 - Tables

Table 1: The Table reports the performance of the models described in section 2.1. All models were trained on the *mini_train_set*, containing 10000 images from the CIFAR-100 dataset, and validated against the *mini_valid_set* containing 3000 images from the same. The validation multi-class cross entropy error reported was computed when validating the performance of the model with the hyper-parameter settings specified in the right column. The results are reported in minimum validation error ascending order.

| **Learning Rule** | **Minimum Validation Error** | **Hyper-parameter Settings** |
|---|---|---|
| Momentum Learning Rule | 3.658750 | $\eta$=0.001<br>$\alpha$=0.7 |
| Simple Gradient Descent | 3.667828 | $\eta$=0.005 |
| Adam Learning Rule | 3.706621 | $\eta$=0.0005<br>$\beta_1$=0.9<br>$\beta_2$=0.9999 |
| RMSProp Learning Rule | 3.735100 | $\eta$=0.0001<br>$\beta$=0.95 |

Table 2: The table reports the performance of all models trained and validated on the CIFAR-100 dataset. All models were trained over 100 epochs (except for *Droupout* run for 400 epochs) on a training set containing 40000 images, each belonging to a 1 of 100 categories. The models were then validated against a set of 10000 images previously held out for validation purposes. The table reports the multi-class cross entropy error and classification accuracy of the models. All algorithm were run with the settings reported on the right-most column. The results are reported in maximum accuracy descending order.

| **Learning Rule** | **Minimum Validation Error** | **Maximum Accuracy** | **Run Time (sec)** | **Hyper-parameter settings** |
|---|---|---|---|---|
| Gradient Descent with Dropout | 3.101271 | 0.249599 | 1122.09 (x4) | $\eta = 0.005$<br>$keep\_probability = 0.5$ |
| Simple Gradient Descent | 3.293267 | 0.245099 | 968.15 | $\eta = 0.005$ |
| RMSProp Learning Rule | 3.296203 | 0.242500 | 1196.01 | $\eta = 0.0001$<br>$\beta = 0.95$ |
| Momentum Learning Rule | 3.232281 | 0.241299 | 1082.70 | $\eta = 0.001$<br>$\alpha = 0.7$ |
| Gradient Descent with Pre-Training | 3.262129 | 0.233999 | 1540.43 (pre-training)<br>+1002.21 (training) | $\eta = 0.005$ |
| Adam Learning Rule | 3.404130 | 0.214699 | 1580.83 | $\eta = 0.0005$<br>$\beta\_1 = 0.9$<br>$\beta\_2 = 0.9999$ |

# Appendix II: CIFAR-10 - Results Summary

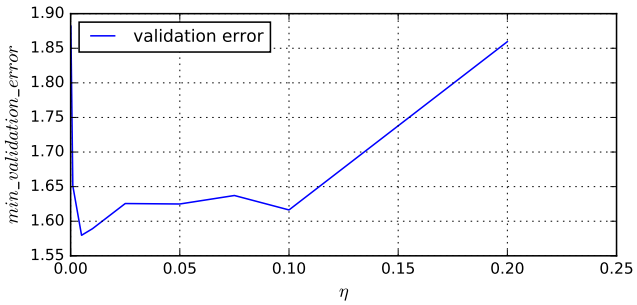
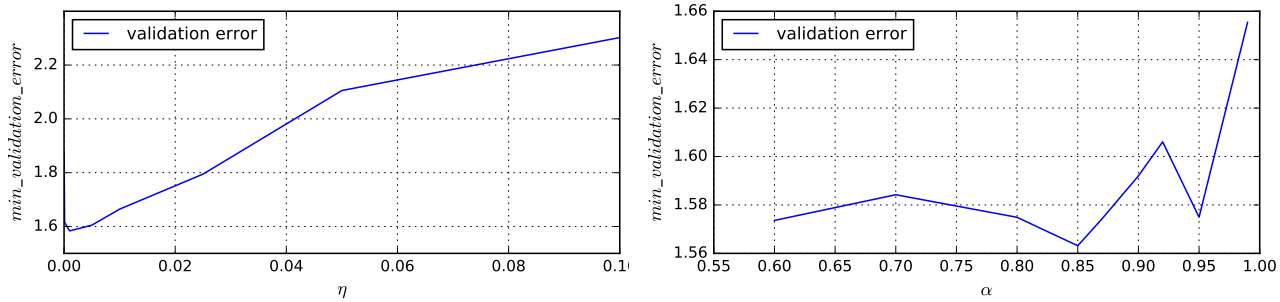


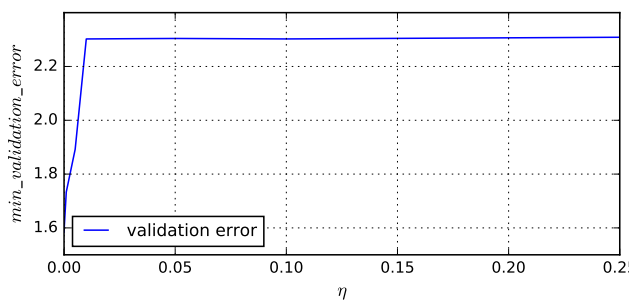

Figure 9: The Figure shows the change in *minimum* validation error of the devised NN when trained with simple Gradient Descent on the *mini_train_set* for CIFAR-10, with varying values of $\eta$. The validation error is computed on the *mini_valid_set* validation set. The value of the hyper-parameter which minimizes the validation error is $\eta = 0.005$.



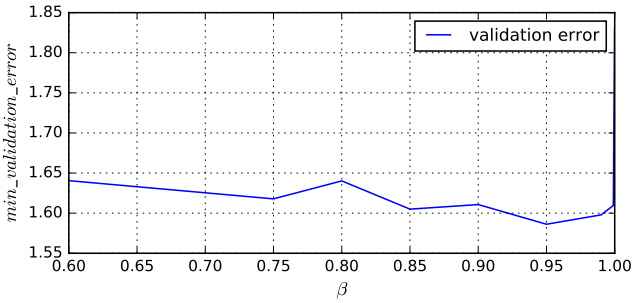

(a) Eta search.

(b) Alpha search.

Figure 10:
Figure (a) shows the change in *minimum* validation error when the NN devised is trained with a Momentum learning rule, fixing the $\alpha$ hyper-parameter to 0.9 (a known a-priory good value) and searching over the $\eta$ space. The value which minimizes the validation error on the *mini_valid_set* set is $\eta = 0.001$
Figure (b) shows the change in validation error when changing the $\alpha$ hyper-parameter while fixing $\eta = 0.001$. The best performance on the *mini_valid_set* for CIFAR-10 is given for $\alpha = 0.85$.

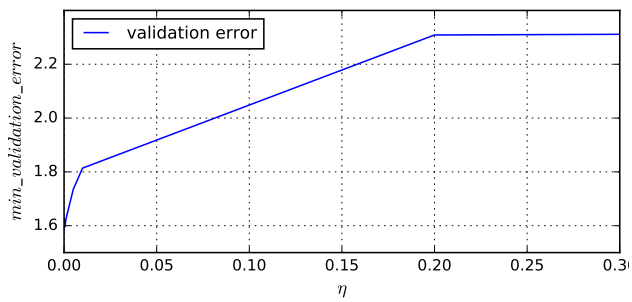



(a) Eta search.

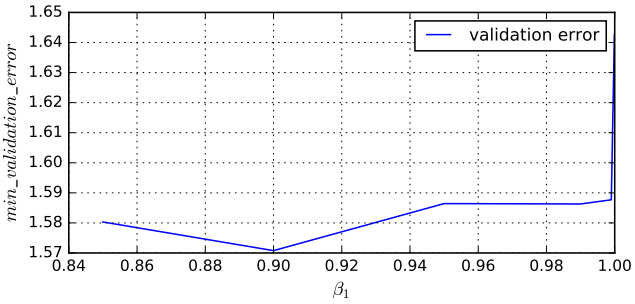



(b) Beta search.

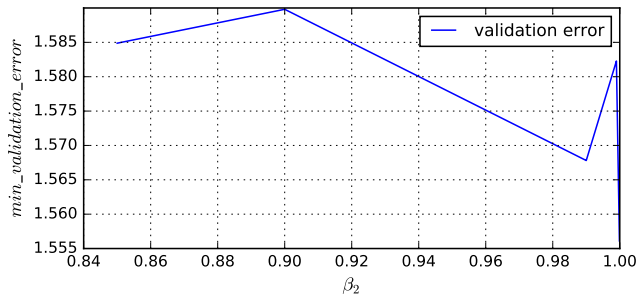
Figure 11:
Figure (a) shows the change in *minimum* validation error when the NN devised is trained with the RMSProp learning rule, fixing the $\beta$ hyper-parameter to 0.9 (a known a-priory good value) and searching over the $\eta$ space. The value which minimizes the validation error on the *mini_valid_set* for CIFAR-10 set is $\eta = 0.00001$
Figure (b) shows the change in validation error when changing the $\beta$ hyper-parameter while fixing $\eta = 0.00001$. The best performance on the *mini_valid_set* for CIFAR-10 is given for $\beta = 0.95$.




(a) Eta search.

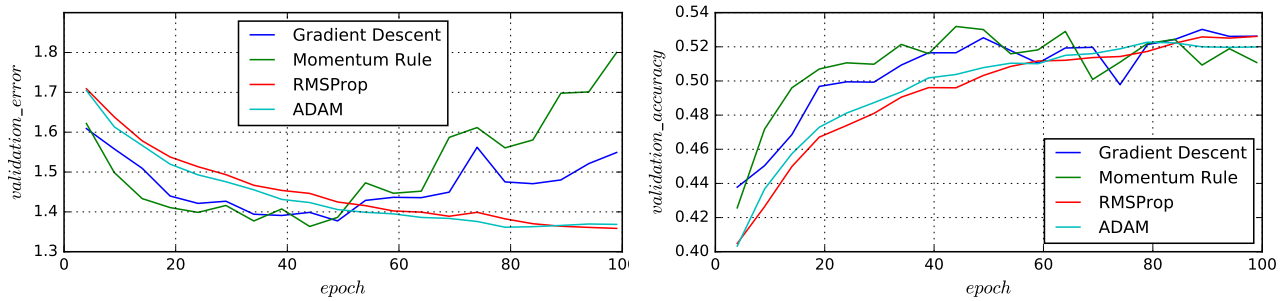



(b) Beta1 search.



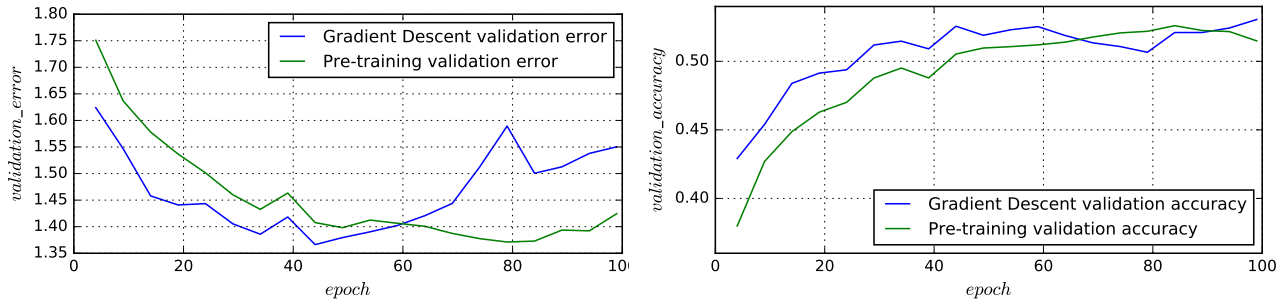

(c) Beta2 search.

Figure 12:
Figure (a) shows the change in *minimum* validation error when the NN devised is trained with Adam learning rule, fixing $\beta_1 = 0.9$ and $\beta_2 = 0.999$ (two known a-priory good values) and searching over the $\eta$ space. The value which minimizes the validation error on the *mini_valid_set* set for CIFAR-10 is $\eta = 0.00001$
Figure (b) shows the change in validation error when changing the $\beta_1$ hyper-parameter while fixing $\eta = 0.00001$ and $\beta_2 = 0.999$. The best performance on the *mini_valid_set* for CIFAR-10 is given for $\beta_1 = 0.9$.
Figure (c) shows the change in validation error when fixing $\eta = 0.00001$ and $\beta_1 = 0.9$, and searching over the $\beta_2$ hyper-parameter space. The minimum validation error on the *mini_valid_set* for CIFAR-10 validation set is reached for $\beta_2 = 0.9999$.



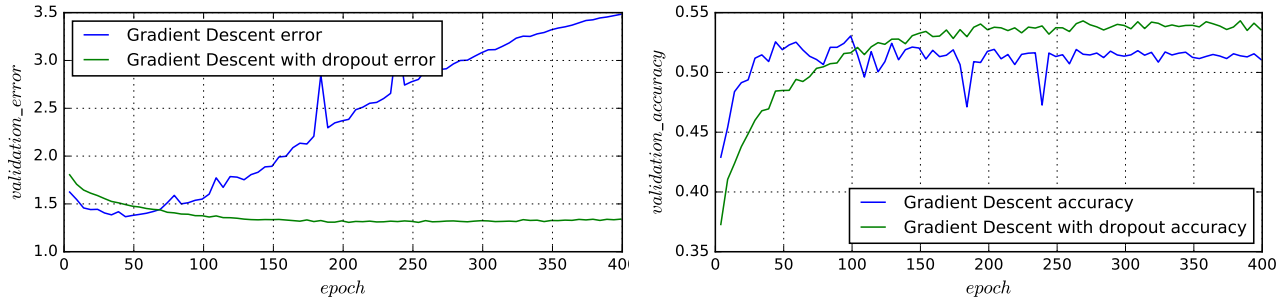

(a) Error (b) Accuracy

Figure 13:
Figure (a) and (b) show the change in, respectively, validation error and accuracy when the NN devised in Section 2.1 is trained with Gradient Descent, Momentum, RMSProp and Adam optimizer over 100 epochs. Each model is trained on the CIFAR-10 dataset with the optimal hyper-parameter settings previously found through a systematic hyper-parameter search.




(a) Error (b) Accuracy

Figure 14: The Figure shows the change in validation error and accuracy of the model trained with Gradient Descent on the CIFAR-10, after its hidden unites were systematically pre-trained to recognize the images in input. The network reaches its best performance sooner than when randomly initialized but does not improve on the baseline top performance of simple Gradient Descent on a network initialized with a GlorotUniform initialization.




(a) Error

(b) Accuracy

Figure 15: The Figure shows the change in Error (a) and Accuracy (b), when training the Neural Network with dropout over 400 epochs through the CIFAR-10 training set, and comparing it to a simple Gradient Descent model. Dropout allows the network to reach a better validation accuracy and error, and shows a better asymptotic behavior over epochs. The introduction of dropout, however, introduces training delays, as $\approx 50\%$ of all the units in the hidden layers are inactive at any given point (thus lengthening gradient training for said hidden units of approximately twice the time).

Table 3: The Table reports the performance of the models described in section 2.1. All models were trained on the *mini_train_set*, containing 10000 images from the CIFAR-10 dataset, and validated against the *mini_valid_set* containing 3000 images from the same. The validation multi-class cross entropy error reported was computed when validating the performance of the model with the hyper-parameter settings specified in the right column. The results are reported in minimum validation error ascending order.

| **Learning Rule** | **Minimum Validation Error** | **Hyper-parameter Settings** |
|---|---|---|
| Adam Learning Rule | 1.556107 | $\eta$=0.00001<br>$\beta_1$=0.9<br>$\beta_2$=0.9999 |
| Momentum Learning Rule | 1.563153 | $\eta$=0.001<br>$\alpha$=0.85 |
| Simple Gradient Descent | 1.579689 | $\eta$=0.005 |
| RMSProp Learning Rule | 1.586194 | $\eta$=0.00001<br>$\beta$=0.95 |

Table 4: The table reports the performance of all models trained and validated on the CIFAR-10 dataset. All models were trained over 100 epochs, except for *Droupout* run for 400 epochs, on a training set containing 40000 images, each belonging to a 1 of 10 categories. The models were then validated against a set of 10000 images previously held out for validation purposes. The table reports the multi-class cross entropy error and classification accuracy of the models. All algorithm were run with the settings reported on the right-most column. The results are reported in maximum accuracy descending order.

| **Learning Rule** | **Minimum Validation Error** | **Maximum Accuracy** | **Run Time (sec)** | **Hyper-parameter settings** |
|---|---|---|---|---|
| Gradient Descent with Dropout | 1.306840 | 0.543200 | 1185.14(x4) | $\eta = 0.005$<br>$keep\_probability = 0.5$ |
| Momentum Learning Rule | 1.363384 | 0.531899 | 1141.09 | $\eta = 0.001$<br>$\alpha = 0.7$ |
| Simple Gradient Descent | 1.377286 | 0.530199 | 1035.26 | $\eta = 0.005$ |
| RMSProp Learning Rule | 1.358536 | 0.526099 | 1255.39 | $\eta = 0.0001$<br>$\beta = 0.95$ |
| Gradient Descent with Pre-Training | 1.371172 | 0.526000 | 1641.15 (pre-training)<br>+1055.22 (training) | $\eta = 0.005$ |
| Adam Learning Rule | 1.361474 | 0.522699 | 1261.72 | $\eta = 0.0005$<br>$\beta\_1 = 0.9$<br>$\beta\_2 = 0.9999$ |

## Appendix III: Coursework 4

After obtaining a simple baseline network for object classification on the CIFAR-100 and CIFAR-10 dataset, the next set of experiments will focus on building more complex networks to improve on the baseline performance of the dropout architecture. The second set of experiments will focus on two architectures: first, we will design a Convolutional Neural Network architecture, and identity a training regiment to obtain the best performance on the dataset; second, we will attempt to improve on the baseline CNN performance through multi-task learning. Multitask learning will be implemented by forcing the algorithm to classify both the 100 object categories previously used and the 20 *super-classes* each incorporating multiple classes. As the network will learn how to recognize both, it will be crucial to devise a method for it to give more weight to the classification of the 100 finer grained classes.