Optimising Facial Keypoint Detection With Deep Learning

Student Name: Alastair Breeze

Supervisor Name: Stephen McGough, Noura Al Moubayed

Submitted as part of the degree of MEng Computer Science to the

Board of Examiners in the School of Engineering and Computing Sciences, Durham University

03.05.2016

***Abstract* —**

Context / Background — A humans visual perception of the world is a skill we are constantly trying to understand in the field of computer vision. In particular, the ability to de-construct an image and analyse keypoints/landmarks is fundamental to many vision problems. Facial keypoints detection, is aimed at detecting facial feature points and has been the focus of significant research efforts thanks to the multitude of applications in areas such as biometric analysis, expression detection and medical diagnosis. In addition, the problem generalises to the further applications of generic keypoint detection.

Aims — The project aims to push current computer vision boundaries with the application of deep learning techniques to optimise the facial keypoints problem. Deep learning has become a popular technique for representing high level abstractions in data through the layering of complex algorithms. Our solution shall train a deep neural network with supervised learning from a labelled training set to accurately predict keypoint locations of unlabelled data with high precision, validating deep learning as a viable model to the problem.

Method — Deep learning has become a powerful branch of modern machine learning, prominent in computer vision problems due to its impressive pattern recognition performance. With advancements in general purpose graphics processing unit (GPGPU) computing, we create a large convolutional neural network (CNN), applying state of the art deep learning optimisations and exploring data pre-processing enhancements and augmentations. We develop a novel algorithm for visualisation heat-maps attempting to directly understand and improve the model.

Results — We have trained and optimised a range of CNN configurations to develop a model that can predict at a high precision against an error function. The results are benchmarked against models from data scientists from around the world, validating the successes of our model. We found that regularisation techniques have shown promise, whilst others showed minimal improvement. Our visualisation heat-maps validated our model, with potential shown for applications of such an algorithm.

Conclusions — A CNN model has been demonstrated as a powerful model for the facial keypoints detection problem, performing well in our tests and against researchers around the globe. Our model is fine-tuned for the generic case, but struggles with outliers that differ from the classic dataset. We outline ideas for future scope in the problem, discussing potential research areas that could enable the next level for facial keypoints detection.

***Keywords* —** Computer Vision; Facial Keypoint Detection; Deep Learning; Machine Learning; General Purpose Graphics Processing Unit (GPGPU); Convolutional Neural Network (CNN); Regularisation

I INTRODUCTION

Computer vision has consistently driven artificial intelligence (AI) research as a means to test and benchmark machine intelligence. In recent years we have made huge leaps forward in the field of machine learning (ML) thanks to the advancements in the sub-field of deep learning. Deep learning has become a hot topic based around building algorithms that can represent high level abstractions in data through algorithmic layering. Such high level abstractions have enabled researchers to represent higher level features and patterns in neural networks at less computational cost at improved accuracy than ever before. An enticing feature of deep neural networks is how, once trained, it can propagate predictions fast, making it popular for intelligent systems.

In this work we investigate the facial keypoints problem: Given an image of a face, pinpoint the locations of particular feature keypoints (sometimes referred to as landmarks) eg. tip of the nose, centre of the eye. A competition was formed by Kaggle, a data-science competition organiser, to standardise this problem[1], allowing researchers to benchmark approaches against other researchers from around the world. The Kaggle dataset contains images of both labelled and unlabelled keypoints data, for example see in Figures 1(a-g) input samples. Using the labelled data, the system can be trained to predict labels for the test data set, for semi-marking by the competition[2] against a root mean squared error (RMSE) objective function:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{1}$$

where $\hat{y}$ is the predicted value and y is the ground truth, with n as the number of sample points.

The difficulty of the challenge lies in the uncertainty that follows with each image, where each can yield hidden challenges like orientation, scale, image quality, facial obstructions or generally the facial expression in question. The system needs to be robust enough to be capable of dealing with such extreme examples. In addition to this, we have a relatively small dataset of roughly 2000-7000 training examples for each feature in comparison to other deep learning applications that can have of order millions of training data due to the nature of their sources.

A Potential Applications

Keypoints are fundamental to what make up vision as we know it. They can be defined on any objects and have applications in an array of problems. Through research into the facial keypoints detection, researchers hope to unlock secrets that can be re-applied to the general problem. Specifically, facial keypoints detection yields potential applications in real world problems:

- **Emotion Tracking —** Knowing facial keypoints unlocks insight into a person's emotion. Research by companies like Microsoft have applied such research to alarm clocks where a specific facial emotion must be made to unlock the phone, proving consciousness[3].

[1] 'Kaggle Facial Keypoints Challenge', `https://www.kaggle.com/c/facial-keypoints-detection` (2016)

[2] A process of marking 50% of the submission until the end of the competition to add uncertainty.

[3] Microsoft Project Oxford `https://www.microsoft.com/en-us/garage/workbench-apps-details/Mimicker-Alarm/MimickerAlarm.aspx` (2016)

- **Biometrics/Facial Recognition —** Facial keypoints are core to what make up an identity. A system could be developed to 'fingerprint' a persons keypoints as a biometric system or in cohesion with modern facial recognisers, adding additional levels of security like unique facial expression to authenticate.
- **Medical Diagnosis —** In medicine, facial keypoints could be used in stroke detection. A common early sign of a stroke is facial features becoming malformed. Every second is vital, and being able to detect the early signs can significantly reduce damage with some recent unpublished research by the Hong Kong Polytechnic University[4] claiming to have made capable systems in the area.

B Project Aims and Deliverables

The project tackles the research question: 'How well can we optimise the facial keypoints problem using a CNN model with state of the art regularisation techniques and developing new algorithms?' We set out to satisfy deliverables in Table 1, we set out to achieve project aims:

1. **Validating** a CNN as a viable model for the facial keypoints problem.
2. **Optimising** the CNN model to competitively challenge researchers from around the globe.
3. **Exploring** optimisation, regularisation and visualisation methods to perfect our model.

Table 1: Project Deliverables

Deliverable	Type
Data Management System — a system that efficiently reads the training & testing data into relevant data-structures and memory	Minimum
Deep CNN Trainer — a system that can use supervised learning to train a CNN	Minimum
Checkpoint Management System — a storage system that can manage program states	Minimum
Prediction System — a system that can load a trained network state to make keypoint predictions on input data	Minimum
Pre-Processing System — an image processor that enhances and unifies data	Intermediate
Data Augmenter — artificially expanding our given datasets, or introducing new datasets in attempt to broaden our training data	Intermediate
Output Management & Visualisation System — an interface to analyse performance visualisations	Intermediate
Deep Restrictive Boltzmann Machines (DRBN) Pre-Trainer — a system that can pre-train the network before fine-tuning as an initialisation technique	Advanced
Live Stream Tester — A web-cam based program to real time test the system	Advanced

II RELATED WORK

In this section we discuss the current facial keypoints problem research, an increasingly popular problem in recent years since the standardisation of the Kaggle challenge[1]. We then expand our literature review into generalised deep learning research including architectures, optimisations

[4] 'Novel computer intelligence system for acute stroke detection.' `www.sciencedaily.com/releases/2015/05/150512112341.htm` (2016)

and visualisation techniques that forms the base of our model.

A Facial Keypoints Problem

We begin with the analysis of facial keypoints detection systems which do not utilise deep learning, namely (Belhumeur et al. 2011, p.545) that standardised a dataset known as Large Face Parts in the Wild (LFPW) dataset (now part of the Kaggle[1] challenge). Belhumeur et al. discussed a support vector machines (SVM) approach with moderate successes, but most interestingly an optimisation technique used was synthesising and expanding the dataset by rotations, a common technique we use later in this section.

Initial research on a deep learning approach to the facial keypoints problem was from (Nouri 2014). Nouri investigated a CNN architecture built using the open source Lasagne module[5], a Theano (Bergstra et al. 2010, Bastien et al. 2012) based Python module. The approach using a regularised LeNet-5 model from (Lecun et al. 1998). The module abstracts complexity away from the developer, something we set out to avoid.

In 2015, the Int. Conf. Image Processing witnessed an influx of papers on the facial keypoints problem including (Kimura et al. 2015, p.3) investigating mini-batching techniques including augmentations to help expand the dataset, showing positive results addressing the small dataset issue. Image augmentations were a common theme, also mentioned by (Yamashita et al. 2015) who similarly used rotations and scaling on their training set.

B Deep Neural Network Architectures

We broaden our search into general deep learning research, beginning with classic ML computer vision problem MNIST: classifying numeric digits $[0..9]$, given a labelled dataset[6] of $70,000$ examples. (Lecun et al. 1998) played a key part in the research problem through the popularisation of a CNN model, a common component in deep learning architectures today that enables translation invariant feature detection, a component that laid the foundation to our model.

ImageNet is a modern, and very much unsolved and expanding, challenge (Russakovsky et al. 2015). The annual challenge entails problems like object classification or description generation and hard problems that require the most powerful deep learning architectures we know today. An early ImageNet contender was (Krizhevsky et al. 2012), their work uses a CNN model with rectifier linear unit (RELU) activation functions as a means to accelerate learning. It claims to accelerate learning to a defined error by roughly 6 times, a characteristic we experimented in our network. Recent ImageNet submissions include the 'GoogLeNet' (Szegedy et al. 2014), a radical approach to CNN architectures that use branching streams of differing convolutions, an approach producing powerful results thanks to it's scale invariant convolutions. A more classic ImageNet submission is the OxfordNet/VGG Net (Simonyan & Zisserman 2014), a model we took inspiration from in our architecture. The model uses a seemingly classic CNN architecture whilst pushing the levels of depth to superb performance.

[5] 'Lasagne', `http://lasagne.readthedocs.org/en/latest/` (2016)

[6] 'The MNIST Database of handwritten digits', `http://yann.lecun.com/exdb/mnist/` (2016)

C Optimisation Research

Regularisation algorithms optimise our model and help prevent over-fitting, a problem when a system fits the training data over unseen testing examples. Microsoft researcher (Bottou 2012) draws on a selection of different neural network optimisation techniques. One of which is the rate at which we traverse the state space, known as the learning rate. Bottou discusses dynamic learning rates that change as time goes on to help accelerate to a minimum error.

Another common technique is dropout (Srivastava et al. 2014), this technique relies on natural phenomena to optimise the network by training on random subsets of the neurons and praised by many (Krizhevsky et al. 2012).

Early stopping is the process of stopping training before the network over-fits the data. In the Theano documentation[7], the developers use a patience concept, however this syntax soon became unnatural to understand. (Prechelt 2012, p.2) discuss early stopping in a simpler fashion, an approach we build on by simply stopping when we no we can no longer consistently reduce error against our validation set.

Network initialisation is fundamental to finding a state that will converge using a stochastic gradient descent (SGD) training algorithm. (Glorot & Bengio 2010) formalised the industry accepted method for TANH activation functions, but a debated discussion with regards to the RELU activation function. A further step from the random initialisation includes using deep restrictive Boltzmann machine (DRBM) to pre-train the network in an unsupervised manner to statistically establish our features before fine tuning by our classic SGD model. The model relies on built up restrictive Boltzmann machines (RBM) stacked together to mirroring the core network weights. (Hinton 2006) pioneered DRBM research, a technique that was applied to keypoints challenge by (Haavisto 2013) pre-training followed by supervised fine tuning. The approach however suffers when compared to other Kaggle entries, limited by it's architectural simplicity. Stanford University (Lee et al. 2009) investigates extending DRBM's to CNN models that could potentially build onto our complex CNN model see Figure 1.

Another pre-training technique was also explored by (Yamashita et al. 2015), who investigated using a simplified version of the problem to classify parts of the face where the network can then be transferred as the starting state for the fine-tuned keypoint analysis.

D Visualisation Techniques

Visualising deep neural networks allow us to understand the hidden logic to the network's intelligence. One of the best visualisation papers was (Zeiler & Fergus 2013), where novel techniques were explored for the visualisation of convolutions. The occlusion technique has grown popular in the research community, used by Google (Weyand et al. 2016) on a location classification problem and Facebook (Sun et al. 2015) in the ImageNet object classification and localisation problem. Google, like Zeiler and Fergus, implements the simple form of the algorithm, while Facebook uses a more refined approach that has great success in localising objects in images.

[7] 'Deep Learning Tutorial', `http://deeplearning.net/tutorial/deeplearning.pdf` (2016)

Little detail into the algorithms are published, and thus is an avenue we shall further define and justify formally. We investigate ways to apply it back into the model as an optimisation technique.

A key observation in keypoint recognition by (Hou et al. 2015), was that you don't necessarily use the whole image in order to pinpoint a keypoint best. Hou implemented a cascading network where a prediction to relocate the zone of the image, and recurring until we are satisfied with the keypoint confidence. Similarly, we will be attempting to use our visualisation system to probabilistically calculate each pixel's importance, and maximising our model around this.

III SOLUTION

Our solution is composed as a compilation of optimisations to maximise our model prediction accuracy. We begin outlining our base model, deriving optimisation techniques, finished by defining our testing and visualisation system.

A Deep Learning Foundation

In this sub-section, we discuss a base model that meet our minimum deliverables from Table 1.

A.1 Deep Neural Network Architecture

Our final model consists of 15 connecting layers shown in Figure 1. By building on the work of (Lecun et al. 1998), we introduced an additional convolutional layer to maximise depth for more advanced and complex pattern detection (Simonyan & Zisserman 2014).

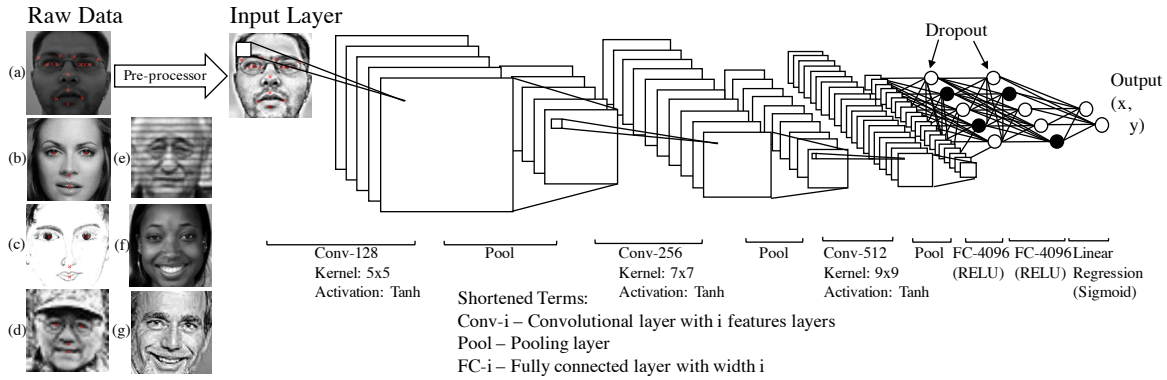


Figure 1: CNN architecture of our solution

Fully Connected Layers (Hidden and Linear Regression) — A simple neural network model taking the form of Equation 2(a), using activation functions of form Equation 3(a-c).

$$(a)\ y = \text{ACTIVATION}(Wx + b); \qquad (b)\ y^k = \text{ACTIVATION}((W^k * x) + b^k), \tag{2}$$

where x is size n, y is size m, W is a $n \times m$ weight matrix, b is a vector size m. The second function, as described by Theano[7], differs in that we are in higher dimensions of multiple feature kernels k that convolve. for different convolutional features. ACTIVATION can take the form

using x as an input neuron:

$$(a)\ \text{SIGMOID}(x) = \frac{1}{1+e^{-x}}; \qquad (b)\ \text{RELU}(x) = \text{MAX}(0,x); \qquad (c)\ \text{TANH}(x) = \frac{1-e^{-2x}}{1+e^{-2x}}, \tag{3}$$

Convolutional Layers — We convolve different feature kernels to the images, combining three architectural ideas to ensure degrees of shift, scale, and distortion (Lecun et al. 1998). Each convolutional layer has a defined number of kernels of certain size. Each different kernel convolves TODO

A.2 Model Training

We employ a mini-batch Stochastic Gradient Descent (SGD) algorithm as a means to supervised training our network, described by (Bottou 2012):

$$w_{t+1} = w_t - \mu \frac{1}{n} \sum_{i=1}^{n} \nabla_w Q(z_i, w_t), \tag{4}$$

with w_t representing the weight parameters at time-step t, μ is the learning rate, z_i is the ith training example, $Q(z_i, w_t)$ is the loss function given the input at given parameters; in our case it is the RMSE from Equation 1. $\nabla_w Q$ is the gradient of the loss function with respect to the network parameters w. The function is averaged over the n training examples, following that with a sufficiently small learning rate μ the algorithm will achieve linear convergence.

The SGD algorithm is a flavour of the gradient descent, where each iteration estimates the gradient based on a randomly picked example z_t rather than computing the gradient averaged over all training examples. It can also be noted that it is possible to extend the SGD as a randomised mini-batch algorithm by picking a small subset to average over z_t:

$$w_{t+1} = w_t - \mu_t \nabla_w Q(z_t, w_t), \tag{5}$$

where μ_t represents a potentially dynamic learning rate of the algorithm at time t. The rate at which we change our network parameters to descend to a local minimum is fundamental to training a network. Too high and we risk potentially skipping a minima, too low and our network will converge too slow. (Bottou 2012) discusses a dynamic learning rate of the form:

$$\mu_t = \frac{\mu_0}{1+\mu_0 \lambda t}, \tag{6}$$

with t representing time (iteration) and μ_0 the initial learning rate, λ are constants. We begin at a high learning rate in order to traverse the state space at a fast rate, then gradually slow the rate in order to focus in on an optima parameter set for the problem.

A.3 Network Initialisation

Ideally we want a network initialisation that will converge to a minimum RMSE in the quickest time, where choices of method is dependent on the activation function. (Glorot & Bengio 2010)

showed that for TANH activation functions, random initialisation performs well in the range:

$$\text{XAVIER RANGE} = \left[-\sqrt{\frac{6}{fan_{in} + fan_{out}}}, +\sqrt{\frac{6}{fan_{in} + fan_{out}}} \right], \tag{7}$$

given fan_{in}, fan_{out} are the number of input neurons and output neurons respectively.

Universally accepted for TANH activation functions, however some of our model uses RELU from Equation 3(b). RELU has been subject to discussion in the research community over initialisation, typically following a similar XAVIER RANGE found to also converge cleanly.

DRBM could have been used pre-train the networks (Hinton 2006), however upon implementation we found this was outside of the scope of the project with anticipated gains being trivial improvement. It would also affect the complexity of our model and thus require us to choose a simpler architecture like (Haavisto 2013), that could introduce a higher error into the model.

B Regularisation Techniques

Avoiding over-fitting the model is ensuring representation of the general case over the training data and is a heavily researched problem area. In this subsection we explore some of the algorithms that help to accomplish to build on-top of our foundation from Section III.A.

L1 & L2 Regularisation — The most common techniques are L1 & L2 regularisation where we add a value with respect to the parameter weights to the cost function, discussed by (Bottou 2012). The clearest description comes from researchers behind Theano[7] where the technique encourages smooth network mappings by penalizing large parameters values, aiding to decrease the non-linearity that the network models.

Dropout — Natural phenomena is commonly found in optimisation techniques, dropout being a key example of this. (Srivastava et al. 2014) discusses the idea and show its successes and is a heavily used regularisation in the deep learning community. The method works by taking input neurons X, and for every neuron outputting the value with a probability p, and 0 with $1-p$ as shown in Equation 8 where X is the n dimensional input with $(i_1, ..., i_n)$ as the n-dimensional coordinate and p the probability of dropping a neuron. We implement this technique on the output of every convolutional and hidden layer in our model.

$$\text{DROPOUT}(X, p)_{(i_1,...,i_n)} = \begin{cases} X_{(i_1,...,i_n)}, & \text{if RAND()} < p \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

Early Stopping — Simplicity was key when choosing an early stopping algorithm, building on (Prechelt 2012, p.2) we opted for a method that tracks when the network is no longer making improvements every k epochs. We trivially create a refreshing counter that monitors this.

C Data Optimisation

Input data strongly influences the model we train. We discuss some of our pre-processing techniques that enabled us to enhance and augment images to maximise the training accuracy.

C.1 Image Pre-Processing

Improving image quality and reducing noise helps make keypoints clearer and therefore easier to predict.

Compression Artefact Removal — Some of the images suffer from block areas known as compression artefacts incurred through lossy compression algorithms like JPEG. By using discrete frequency transform (DFT) and then filtering, we can remove the high frequency details (Solomon & Breckon 2010, pp.135–139) making up the artefacts. Clear improvement is seen in Figure 2(a), by performing the algorithm before sharpening to avoid artefacts amplification.

Smoothing — Reducing noise in non-edge regions of an image can help draw attention away from the non-edge areas and to enhance the actual edge detection. We employed a bilateral filter (Tomasi & Manduchi 2000) to smooth skin areas whilst retaining edge definition as shown in Figure 2(b).

Sharpening — Deep learning is essentially pattern detection using edges as the basis. We enhanced edges through unsharp filtering (Solomon & Breckon 2010, p107) as demonstrated in Figure 2(c).

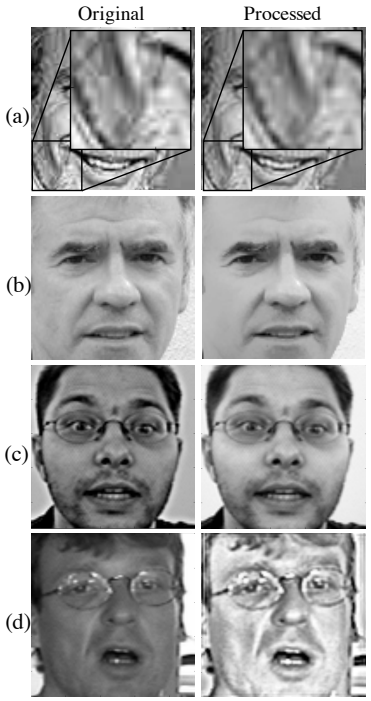


Figure 2: examples of different image processing techniques before and after

Equalisation — The next step of the pre-processing is equalisation, trying to bring the contrast and brightness into a standard uniform range, dealing with under-exposed and over-exposed images. We opted for a contrast limited adaptive histogram equalisation (CLAHE) (Solomon & Breckon 2010, p.78). Figure 2(d) demonstrates the effect of this algorithm.

C.2 Maximising and Expanding the Dataset

Ensuring the best use of our limited dataset is a vital part to our system. We now discuss the techniques and algorithms used to allow us to achieve this.

Maximising Feature Data — A twist of the competition dataset[1] is that some of the input data contains incomplete labels. A trivial solution to this used by (Nouri 2014) to discard images that don't provide a full set of data, an approach that instantly cuts potential data from roughly 7000 to around 2140 examples for certain features. The data originates from sources: BioID Face Database[8] and the LFPW (Belhumeur et al. 2011) dataset. As LFPW is only capable for four features, that dataset is discarded. We split each facial feature into sub-problems, by separating the problem, we maximise the number of data samples for each feature.

[8] 'BioID Face Database', `https://www.bioid.com/About/BioID-Face-Database` (2016)

Facial Symmetry — Facial symmetry is a neat characteristic we can exploit in our system initially explored by (Nouri 2014). We notice training the left eye is the same as training the mirror of a right eye, which can be used for both training set expansion and also predictions. A similar approach can be taken with other symmetric features and can roughly double our training data per feature, reducing processing efforts in doing so.

Augmentations — Artificially expanding our dataset is a simple way to increase size through augmenting the input with image manipulations such as rotations and scaling. We focus on rotations (Kimura et al. 2015), augmenting each training image through rotations around centre:

$$\textsc{RotCoord}((x,y),r) = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} (x-\frac{w}{2})\cos(r) - (y-\frac{h}{2})\sin(r) + \frac{w}{2} \\ (y-\frac{h}{2})\cos(r) + (x-\frac{w}{2})\sin(r) + \frac{h}{2} \end{bmatrix}, \tag{9}$$

given r as an angle in radians and w, h are the width and height of the image respectively.

We experimented with algorithms that attempt to utilise the random rotations nature of our dataset. We predict the keypoint at multiple rotations forming a consensus over our results.

$$\textsc{PredictUniformConsensus}(X) = \frac{1}{2\phi}\sum_{r=-\phi}^{\phi} \textsc{RotCoord}(\textsc{Cnn}(\textsc{RotImage}(X,r)), -r), \tag{10}$$

$$\textsc{PredictGaussianConsensus}(X) = \frac{1}{2\pi\sigma^2}\sum_{r=-\phi}^{\phi} e^{-\frac{r^2}{2\sigma^2}} \textsc{RotCoord}(\textsc{Cnn}(\textsc{RotImage}(X,r)), -r), \tag{11}$$

given ϕ as the rotation range $[-\phi, \phi]$ in degree steps. σ as the Gaussian distribution parameter.

D Visualisation and Validation Systems

Deep learning systems have extensive number of outputs for analysis including logs, tables, prediction visualisations, graphs and more. We handle such large amounts with web data management system that can visualise our outputs in an interactive fashion such as graphs, images. In this sub-section we discuss some of our advanced visualisations we output.

D.1 Censor Heat-map Visualisations

Visualising performance of a deep neural network is a vital component in the optimisation process development. Research has been made in the field (Zeiler & Fergus 2013), but the proposed method aims mainly at the analysis of features with respect to classification problems, where we are performing a linear regression. We propose a more advanced algorithm that builds on a simple occlusion method by Zeiler and Fergus with potential to be applied to a range of problems. We refer to our approach as a censoring algorithm due to nature of our occlusion techniques.

Let our input image be X; where we wish to know the importance of pixel (x, y). We

calculate using $\text{OCCLUSION}(X)_{(x,y)}$ from Equation 12, that can be applied to a heat-map.

$$\text{OCCLUSION}(X)_{x,y} = \sum_{\sigma \in Q} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{2\pi\sigma^2} e^{-\frac{(x-i)^2+(y-j)^2}{2\sigma^2}} |\text{CNN}(\text{CENSOR}(X, i, j, \sigma)) - \text{CNN}(X)|, \tag{12}$$

using $\text{CNN}(X)$ as a black box function that predicts keypoint coordinate, in our case this is a convolutional neural network. σ's are defined from the list Q. $\text{CENSOR}(X, i, j, \sigma)$ is our abstract censoring function that will return an occluded image of X at censor coordinate (i, j) with a σ as the width of the censor:

$$\text{CENSORNOISE}(X, i, j, size)_{x,y} = \begin{cases} \text{RAND}(), & \text{if } i < x < i + size, j < y < j + size \\ X_{x,y}, & \text{otherwise} \end{cases} \tag{13}$$

$$\text{CENSORGREY}(X, i, j, size)_{x,y} = \begin{cases} \alpha, & \text{if } i < x < i + size, j < y < j + size \\ X_{x,y}, & \text{otherwise} \end{cases} \tag{14}$$

given α as an arbitrary grey value, typically 128.

$$\text{CENSORBLUR}(X, i, j, size)_{x,y} = \begin{cases} \sum_{u=-\mu}^{\mu} \sum_{v=-\mu}^{\mu} X_{(x+u),(y+v)} \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}, & \text{if } \substack{i<x<i+size \\ j<y<j+size} \\ X_{x,y}, & \text{otherwise} \end{cases}, \tag{15}$$

with σ representing a value for the Gaussian, μ is the size of the Gaussian filter.

D.2 Live Stream Testing System

Real time evaluation is a powerful tool that allowed us to evaluate the system, useful for observing the systems reactions. We implemented a UDP based client-server model to be run on a GPGPU cluster as it becomes infeasible loading 8 networks each of roughly 300MB GPGPU memory. The initial facial detection was performed through a simple cascade classifier (Szeliski 2011, pp.658–668), passing greyscale cropped faces to the GPGPU cluster server.

E System Implementation

Our solution is based on the Theano programming library (Bergstra et al. 2010, Bastien et al. 2012), a highly optimised Python ML library. Python allowed us to implement fast data manipulations for example augmentations and image processing, in cohesion with all the Theano's highly GPGPU optimised ML functions.

IV RESULTS

Optimisation techniques require tuning to maximise and justify their usefulness. To do so, we investigate parameter tuning with methods that allow tuning multiple parameters: grid search; randomised grid search (Bergstra & Bengio 2012). Results were conducted with Python and Theano[9] through submission to a high performance GPGPU-cluster[10].

[9]Python: 2.7.7, Theano: *0.7.0.dev-RELEASE*

[10]4× *Intel Xeon CPU E5-2650 v3 @ 2.30GHz; 64GB DDR3 RAM;* 2× *Nvidia Tesla K40c GPUs, 12GB GDDR5 each, 2880 CUDA cores each*

A Keypoint Detection

Samples of the predictions of our model are shown in Figure 3. Sub-images (a,b) display near perfect predictions, despite the different orientations. Figure 3(c) also shows a low error, despite orientation and occluding glasses. An obstructing fringe in (d) caused a high level of error. This shows the model tends to overfit their eyebrows which are likely to be the cause of the poor result.

(a) (b) (c) (d) (e)

Figure 3: trained CNN predictions examples. Testing Data (a-d); Live Stream Tester (e)

Live stream testing[11] is demonstrated in Figure 3(e) showing a close prediction to the reality despite obscure facial expression/morphing. The solution achieved nearly 1 frame per second, bottlenecked mainly by the GPGPU memory loading speeds. This shows that the solution does have the capability to be extended to the applications discussed in Section I.A.

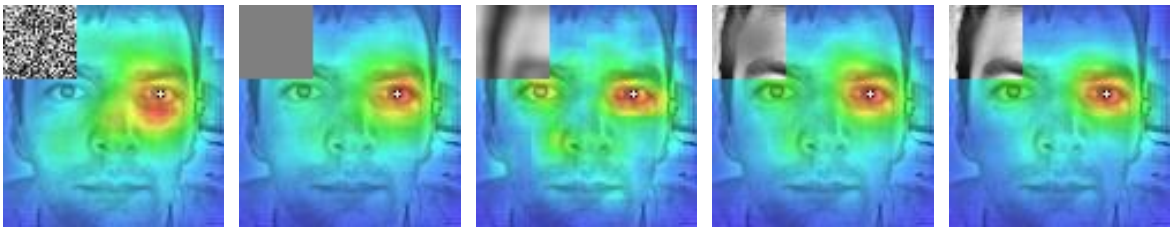
B Censor Heat-Map Visualisations

Having refined the algorithms involved in the heat-map generation, we now test different configurations. In Figure 4(a-e) we consider the 5 techniques CENSORNOISE flat, CENSORGREY flat, CENSORBLUR flat, CENSORGREY faded, CENSORBLUR faded respectively. It is immediately clear that the CENSORNOISE creates a large effect zone, confusing the CNN and not localising very closely. The CENSORGREY and CENSORBLUR perform similarly, both displaying the effect of different facial features. We argue that the CENSORBLUR displays higher contrast, a desirable characteristic of a heat-map.

The faded effect against the plain square effect of the censor, gathers slightly smoother and refined heat-maps to features. We expect this due to the effect that a plain square censor implicates a new edge around the censor, not incurred by the faded censor. This novel addition to the (Zeiler & Fergus 2013) algorithm could apply well to localised classification problems like (Sun et al. 2015) where it is also required to localise the classified objects.

Occasionally censoring pixels were found to benefit the model. We attempted to form an algorithm to calculate relative pixels to the feature that probabilistically are better censored. Upon testing the hypothesis we could achieve roughly 0.001% improvement, unfortunately not significant enough for us to claim as a viable optimisation.

[11] Youtube 'Live Stream Tester' `https://youtu.be/Q5xlQd4KeFE` (2016)



(a) (b) (c) (d) (e)

Figure 4: examples of different censoring heat-maps

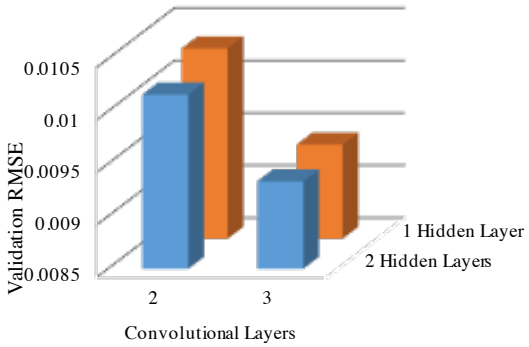


Figure 6: Different architectural depths on the best validation RMSE at 100 epochs

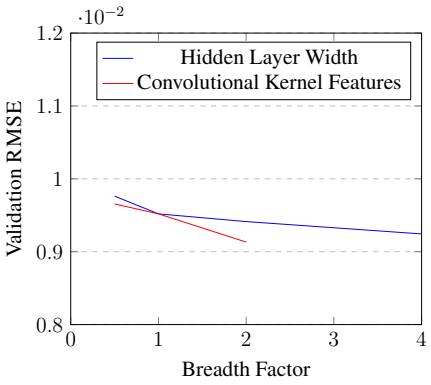


Figure 7: Different network widths on the best validation RMSE at 100 epochs

C *Deep Neural Network Architecture*

Our architecture scale yields configurations that we want to tune to minimise error whilst remaining computationally feasible.

Activation Functions — Core layers to the network architecture required activation functions to their resulting output. We compare SIGMOID or RELU from Equation 3(a,b) in Figure 5, corroborating with (Krizhevsky et al. 2012) where RELU is claimed to accelerate learning.

Depth vs. Breadth — An underlying question when designing a deep learning architecture is depth vs. breadth. (Nielsen 2015, ch4) discusses how a single layer is capable of representing any function, yet it is depth that can represent higher level pattern recognition at reduced computational cost (Sutskever et al. 2013). In our system we have the functional requirement to create a system that can be trained in just a few days.

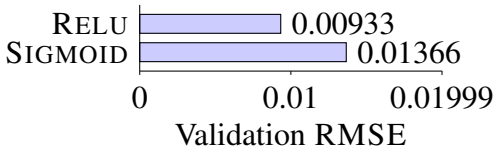


Figure 5: Different activation functions on hidden layers best validation RMSE at 100 epochs

Depth in the convolutional and hidden layers are shown in Figure 6. Due to the nature of the 96×96 input images, 3 levels of convolution and pooling is a sensible and reaches outputs of size $6 \times 6 = 36$ neurons, a manageable number for fully connected layers. The chart emphasises the improvement of 3 convolutional layers over 2 by gaining roughly 8% RMSE reduction. We

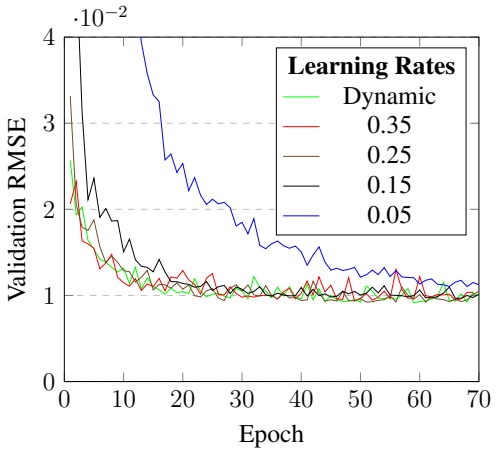


Figure 8: Different learning rates on the best validation RMSE

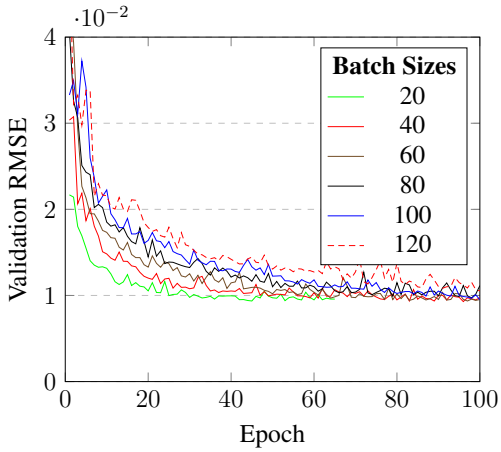


Figure 9: Different batch sizes on the best validation RMSE

test hidden layer depth in a similar way, comparing 1 vs. 2 layers (not including the final linear regression layer). Figure 6 displays lower improvement of around 1%, a welcomed improvement, however we opted for 2 layers to focus computing power elsewhere.

Breadths are visualised in Figure 7. (Szegedy et al. 2014) follows the internal structure we looked to replicate, starting convolutions small progressing to larger and more complex patterns increasing sizes and amounts of our kernels. We consider 3 kernel values: $\{[64a, 128a, 256a] : \forall a \in [\frac{1}{2}, 1, 2]\}$, finding significant improvement at the highest, but at computational cost that makes it infeasible to continue expanding. Figure 7 shows small but gradual improvements as we scale the number of hidden layer neurons. At around factor 4 of the original 1024 neuron widths, we settle at max computable in a day.

D Regularisation Optimisations

Optimising regularisation techniques at parameter configurations are investigated.

Learning Rate — A high learning rate can help with convergence as shown in Figure 8 where it is visible see the early benefits of a learning rate in the range of 0.3. As the network converges, we see erratic fluctuations as the high learning rate struggles to converge to a local minima. It then makes sense to follow a lower learning rate, in the range of 0.1, made possible by a dynamic learning rate following Equation 6 from (Bottou 2012). We configure our learning rate to 0.3 at start: $mu_0 = 0.3$. We reduce to a learning rate of 0.1 at around 50 epochs, entailing $\lambda = -0.4$ and as our starting parameter. The effects can be seen in Figure 8, following a similar descent to that of 0.3, yet demonstrating at a less erratic fluctuations at around epoch 40.

Mini-batch Size — Our mini-batch size is another component that affects speed of learning. Figure 9 visualises clear correlation between lowering batch size and consequent rate of learning. When the batch size is 20, the network converges fast resulting in being unable to sustain learning and triggering early stopping. Instead we want a gradual learning rate, more like those of 40 and 60. For this reason we choose 50 as our desired mini-batch size.

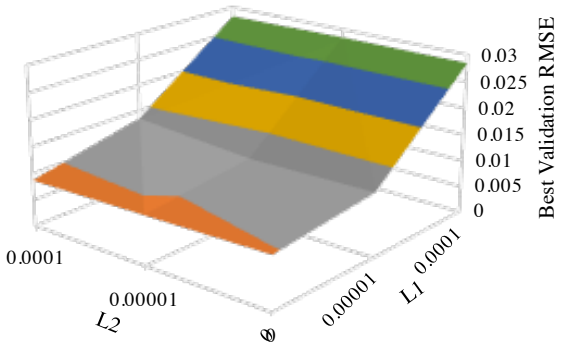


Figure 10: A 3D surface plot of L1, L2 factors using grid search against best validation RMSE at 70 epochs

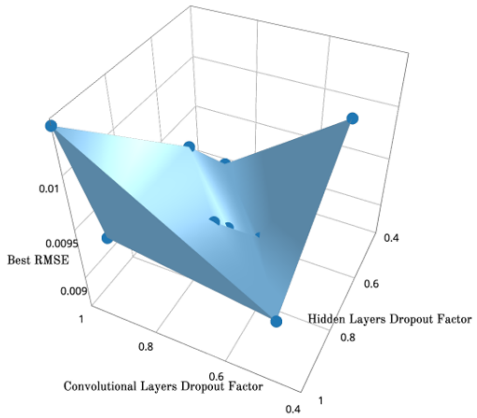


Figure 11: A 3D surface plot of dropout parameters using random grid search against best validation RMSE at 70 epochs

L1 & L2 — After experimenting with different L1 and L2 factors in Figure 10, we found L1 to detriment the model. We could potentially have opted for even lower L1 but didn't seem necessary. L2 did show small reduction to the error, finding L2=0.0001 to be a desirable factor.

Dropout — 3D surface plot's allow us to optimise two parameters in unison. Figure 11 represents a randomised grid search (Bergstra & Bengio 2012) performed on the dropout to convolutional and hidden layers. We randomly choose the two parameters between $[0.4 - 1.0]$, finding error minimal at roughly 0.7 for convolutional layers and 0.6 for the hidden.

E Data Optimisation

E.1 Image Pre-Processing

Different pre-processing algorithms entail varied improvements detailed in Table 2. Improvement was found from all techniques, culminating in a nice overall improvement.

Table 2: Comparison of image processing techniques on best validation RMSE at 80 epochs

Technique	Best Validation RMSE	Percentage Improvement
Compression Artefact Removal	0.00970	0.41
Smoothing	0.01001	0.24
Sharpening	**0.00990**	**3.51**
Equalisation	0.01017	1.41
All	**0.00952**	**5.29**

Unsurprisingly, we gain the most reward from the sharpness filter as deep learning is fundamentally about edge detection. This is shortly followed by equalisation, an algorithm that helped bring the images into similar value ranges when under or over-exposed. Compression artefact

removal shows small improvement, helping before sharpening to deal with a problem that is amplified by the sharpening. Smoothing has a smallest effect by cleansing noisy areas like skin, but in some cases resulting in the low improvements demonstrated.

E.2 Augmentations

Rotations could have been randomly distributed. We instead maximise rotation variance of each training datum by duplicating it in a memory manageable 5 orientations: $(-2f, -f, 0, f, 2f)$ for some constant f radians. Figure 12 displays the effect of different f on our model, concluding best validation fit at $f = 4$.

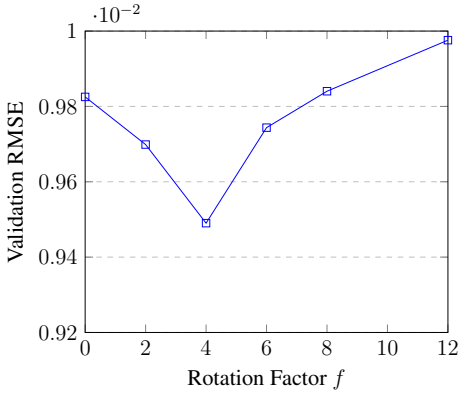


Figure 12: Different rotation factors on best validation RMSE at 90 epochs

Testing our model with augmentations was attempted. Small tests on small subsets of data concluded a small rotations of $[-2..2]$ using a Gaussian consensus function was better than uniform. In testing application we achieved only 0.157% improvement.

F System Performance

GPGPU over CPU performance was unprecedented, Figure 13 exhibiting roughly a $40\times$ speed-up in our system, justifying optimisation claims from (Bastien et al. 2012, p.7).

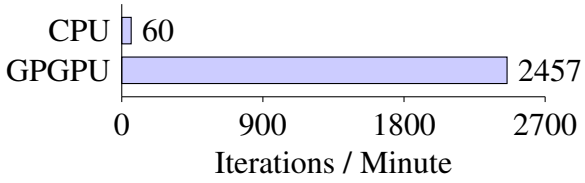


Figure 13: Training example iteration frequency

V EVALUATION

In this section we discuss the strengths and limitations of the system, relating back to the original research question: 'How well can we optimise the facial keypoints problem using a CNN model with state of the art regularisation techniques and developing new algorithms?'

A Strengths

The prediction performance on the model was excellent, achieving 2nd place in the world Kaggle challenge rankings[1](December 2015). We begin by evaluating the strengths that allowed the model to achieve such low error levels.

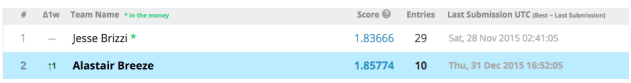


Figure 14: Kaggle leaderboard December 2015

Our modular approach made it easy to manage data, program states forming a strong foundation to meet the minimum deliverables of our project with ease. Convolution was an effective tool in feature detection, where translation invariance proved advantageous for minimising error. We showed that by increasing depth and breadth into our model, we

reduce the testing error but at computational cost to the system.

Regularisation approaches significantly improved the model, with techniques like random search and grid search allowing for easy parameter optimisation (Bergstra & Bengio 2012), and our output management system[12] made it clear analysing output data. Dropout (Srivastava et al. 2014) was the most useful, considerably reducing our models overfitting. Early stopping also tested as a vital method in our algorithm helping to ensure we don't over train the model.

Data optimisation deliverables both helped to regularise the model. Impressive improvements were demonstrated by the training data augmentations helping fit the testing set. Image pre-processing algorithms verified the importance for high quality input data.

Our CNN model is significantly fast to test as demonstrated by the the live stream testing system deliverable, bottlenecked by network latency and GPGPU memory transfers yet capable of achieving a frame per second streaming. This makes it ideally suited to the potential applications defined in Section I A. Our solution is versatile and modularised, making it ideal for re-application to other keypoint problems or more general deep learning. Python was a very powerful language, allowing for great data management and image processing to feed into the model.

We demonstrated an extension to a simple occlusion algorithm by (Zeiler & Fergus 2013). Our approach showed a novel idea that suppresses occlusion induced edges, arguably improving the localisation of the heat-map. This achievement has potential in other deep learning problems due to the improved localisation of the approach.

B Limitations

Our model fails to predict facial keypoints of outlier test example, and by using a RMSE objective function from Equation 1 we amplify large errors into our score. For example, Figure 3(d) demonstrates an inaccurate prediction likely due to the eyebrows being covered by hair.

The high complexity of our model entailed limiting us to not using pre-training techniques such as DRBMs (Lee et al. 2009). (Haavisto 2013) researched DRBMs on a simplified fully connected model with some successes on the keypoints problem but achieving 3.45612 RMSE on the competition, much weaker than the performance of our model at 1.85774 using convolutions. For this reason we explored the possibility of the deliverable, but decided against implementation.

The Kaggle dataset was not perfect, consisting of two different datasets: BioID Face Database[8]; LFPW (Belhumeur et al. 2011) dataset. There exists discrepancy between particular features like the nose and their location on a face, for example where one commonly labels at the tip of the nose while the other at the bottom, introducing an entropy to the best prediction of the system.

By separating the different feature recognitions into their own models, we optimised the quantities of data we have for each facial feature. In doing so we have made the problem roughly 9 times larger to represent, resulting in a more limited approach to the sizing of the network.

[12] 'Project Output Management System' `http://idl.alastairbreeze.com` (2016)

Furthermore, storing the network takes roughly 457MB per feature, equating to 631MB when loaded onto GPGPU memory, a considerably large amount and thus limiting the portability of the solution. Loading all networks requires of order gigabytes of GPGPU memory, concluding the system as inefficient for small devices like mobile phones.

C Approach

An Agile process model was adopted during the project, ensuring incremental developments were made in a weekly cycle, whilst being able to responding to changes with ease. Progress was evaluated in weekly meetings and goals were created that laid basis to the following week.

The modularity of the approach enabled layering of algorithms made easy. We were able to quickly build a base object orientated module, with support for data management solutions. The high complexity of Theano's C++ compiler made for difficult debugging, but generally a well documented and discussed field making for smooth development.

Our output management system[12] allowed for fast system evaluation when testing. Fundamentally, it made possible fast analysis of multiple tests using algorithms like normal and random grid search (Bergstra & Bengio 2012), enabling unprecedented parameter optimisation.

If we were to repeat the work, we would use a higher level ML library, allowing us to develop more optimised, larger and more complex network architecture like GoogLeNet (Szegedy et al. 2014). Since the project, there have been higher level libraries released like Tensorflow Python ML library[13], built for high scaleability and with pre-optimised deep learning architectures.

VI CONCLUSIONS

We have shown that the CNN model to be a powerful approach for the facial keypoints problem through meeting our requirements. We reflect back to our original project aims:

1. We **validated** the CNN model as an excellent approach to the facial keypoints problem.
2. We **optimised** a CNN model and created a system that reached 2nd place in the world on the Kaggle challenge[1](December 2015).
3. We **explored** new optimisations and visualisation techniques with promise for future research avenues, including a novel extension upon the visualisation of a network.

To summarise, we have built upon the work of (Nouri 2014) and expanded a LeNet-5 model (Lecun et al. 1998), increasing depths and breadth (Sutskever et al. 2013). We then explored regularisation optimisations like the work of (Bottou 2012), dropout (Srivastava et al. 2014) to pre-processing and augmentations (Hou et al. 2015) that allowed us to reach such low RMSE.

We explored novel ideas like prediction augmentations and visualisations. Our prediction augmentations built on popular training augmentations (Krizhevsky et al. 2012), now applied to the testing set, showing a minor improvement.

[13]TensorFlow: Large-Scale ML on Heterogeneous Distributed Systems `http://download.tensorflow.org/paper/whitepaper2015.pdf` (2016)

We anticipate future research in the problem to continue improving the model, with aim to deal with extreme examples the system incurs high error. In addition we would like to scale down the network for use on smaller systems. Pre-training algorithms (Yamashita et al. 2015, Haavisto 2013) could be further investigated as they were deemed out of the projects scope.

Our novel heat-map extension has shown great promise for understanding our model. Future work should also be to further validate the approach, by investigating the censoring extension further and applied to other problems.

References

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J. et al. (2012), 'Theano: new features and speed improvements', Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Belhumeur, P. N., Jacobs, D. W., Kriegman, D. J. & Kumar, N. (2011), Localizing parts of faces using a consensus of exemplars, *in* 'In Proc. Int. Conf. Computer Vision and Pattern Recognition', IEEE.

Bergstra, J. & Bengio, Y. (2012), 'Random search for hyper-parameter optimization', *J. Mach. Learn. Res.* **13**, 281–305.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R. et al. (2010), Theano: a CPU and GPU math expression compiler, *in* 'In Proc. Python for Scientific Computing Conf.'.

Bottou, L. (2012), *in* 'Neural Networks: Tricks of the Trade', Vol. 7700 of *Lecture Notes in Computer Science*.

Glorot, X. & Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *in* 'In Proc. Int. Conf. on Artificial Intelligence and Statistics'.

Haavisto, M. (2013), Deep generative models for facial keypoints detection, Bachelors thesis, Lappeenranta University of Technology. unpublished.

Hinton, G. E. (2006), 'Reducing the dimensionality of data with neural networks', *Science* **313**(5786), 504–507.

Hou, Q., Wang, J., Cheng, L. & Gong, Y. (2015), Facial landmark detection via cascade multi-channel convolutional neural network, *in* 'Int. Conf. on Image Processing', IEEE.

Kimura, M., Yamashita, T., Yamauchi, Y. & Fujiyoshi*, H. (2015), Facial point detection based on a convolutional neural network with optimal mini-batch procedure, *in* 'Int. Conf. on Image Processing', IEEE.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in Neural Information Processing Systems 25', Curran Associates, Inc., pp. 1097–1105.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *IEEE* **86**(11), 2278–2324.

Lee, H., Grosse, R., Ranganath, R. & Ng, A. Y. (2009), Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, *in* 'In Proc. Int. Conf. on Machine Learning', ACM, pp. 609–616.

Nielsen, M. A. (2015), *Neural Networks and Deep Learning*, 1st edn, Determination Press.

Nouri, D. (2014), 'Using convolutional neural nets to detect facial keypoints tutorial'.
URL: *http://danielnouri.org/notes/2014/12/17/using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial/ (Accessed: 2016)*

Prechelt, L. (2012), Early stopping — but when?, *in* 'Lecture Notes in Computer Science', Springer Science + Business Media, pp. 53–67.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S. et al. (2015), 'ImageNet Large Scale Visual Recognition Challenge', *Int. Journal of Computer Vision (IJCV)* **115**(3), 211–252.

Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *CoRR* **abs/1409.1556**.

Solomon, C. & Breckon, T. (2010), *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*, Wiley-Blackwell. ISBN-13: 978-0470844731.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: A simple way to prevent neural networks from overfitting', *Journal of Machine Learning Research* **15**, 1929–1958.

Sun, C., Paluri, M., Collobert, R., Nevatia, R. & Bourdev, L. D. (2015), 'Pronet: Learning to propose object-specific boxes for cascaded neural networks', *CoRR* **abs/1511.03776**.

Sutskever, I., Martens, J., Dahl, G. E. & Hinton, G. E. (2013), On the importance of initialization and momentum in deep learning, *in* 'In Proc. Int. Conf. on Machine Learning', AML, pp. 1139–1147.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. et al. (2014), 'Going deeper with convolutions', *CoRR* **abs/1409.4842**.

Szeliski, R. (2011), *Computer Vision*, Springer London.

Tomasi, C. & Manduchi, R. (2000), Bilateral filtering for gray and color images, *in* 'Int. Conf. on Computer Vision', IEEE.

Weyand, T., Kostrikov, I. & Philbin, J. (2016), 'Planet − photo geolocation with convolutional neural networks', *CoRR* **abs/1602.05314**.

Yamashita, T., Watasue, T., Yamauchi, Y. & Fujiyoshi*, H. (2015), Facial point detection using convolutional neural network transferred from a heterogeneous task, *in* 'Int. Conf. on Image Processing', IEEE.

Zeiler, M. D. & Fergus, R. (2013), 'Visualizing and understanding convolutional networks', *CoRR* **abs/1311.2901**.