Face to Age

Project 1
CS395T - Deep Learning Seminar

Aishwarya Padmakumar, Ashish Bora, Amir Gholaminejad

October 9, 2016

A Century of Portraits is a dataset that contains frontal-facing American high school year-book photos with labels to indicate the years those photos were taken [2]. In this project we train classifiers to predict the label, given the image. We used several Deep Neural Network architectures for this task, all of which were finetuned with ImageNet pretraining. With VGGNet architecture, we demonstrate significant improvements in classification accuracy reporting test set accuracy of 67.59% and mean L1 error, as compared to 11.31 % achieved by Ginosar *et al.* [2]. Further, we show some visualizations of the trained model to gain insights into the learned model. The code for this project can be found at https://github.com/AshishBora/face2year.

1 Introduction

Deep Neural networks have been central to large improvements in several visual learning tasks. Feature representations learned by deep convolutional neural networks for image classification on large datasets such as ImageNet [1] have been repeatedly demonstrated to be useful for other tasks [6]. Several downstream applications have also greatly benefited from these representations, either when used directly [9, 10] or with appropriate finetuning [3, 5].

In this project our task is to predict the year when a photograph was taken using only the raw pixel values in the image. We model this as a classification problem with class corresponding to every year between 1905 and 2013, a total of 109 classes. We perform perform 109-way classification using two neural network architectures, AlexNet [4] and VGGNet [8].

2 Method

Finetuning is the approach where in order to learn weights for a new task, instead of random initialization, we start with the values from models trained on ImageNet classification task. Thereafter these weights are changed only slightly with a relatively small learning rate. Finetuning has been applied very successfully in a number of vision tasks [10, 9, 3, 5]. Inspired by this, we attempt the the same approach to solve the problem of predicting image labels.

An important choice that we have to make is the form of predictions and an appropriate loss function. We tried several different approaches for the same, which we describe next.

The simplest approach is the following: The last layer predicts one real number. We then Use L1 loss between this prediction and the ground truth as the loss. This is a direct approach since it directly regresses on the objective function. This is also one of our metrics used to judge the model and hence this approach tries to directly improve the L1 metric. But our experiments suggest that this approach does not work very well. For several learning rate values and even after 30000 iterations, we could not get training accuracy any better than 0%. We thus abandoned this approach.

Another simple formulation is to do 109-way classification where each year present in the training set is a separate class, with cross entropy loss. We add a regularization term and call this $loss_1$. This loss is lucrative since it is very similar to the original ImageNet classification tasks. Thus one might expect that the feature representations learned by those models might be well suited for the cross entropy loss. This approach however ignores the inherent structure in the output space, i.e. it does not use known relationships between the classes.

The natural ordering of years imposes inherent structure within classes. This leads to a natural observation – predicting an incorrect year near the target year is less undesirable than predicting one far from the target year. Thus, to use this information, we add a term corresponding to L1 loss. We call the resulting loss function $loss_2$. We could have used the class with maximum probability for computing the L1 loss, but notice that this will lead to zero gradient most of the time since max over logits does not change often. Instead to get good gradients, we use a weighted combination of all the classes, where the probabilities are given by the softmax. This formulation gives gradients to all the classes, and encourages large probability mass near the target.

2.1 Formal Description

In this section we set up some notation and formally describe the loss models that we use.

Let $\{(x_i, y_i)\}_{i=1}^m$ be the training set where x_i denotes the i-th input image and y_i is the corresponding label. Let C be the number of output classes.

We experimented with AlexNet and VGG architectures. In both cases, for a given input x_i, our model outputs $\widehat{y}_i \in \mathbb{R}^C$. We require that this be a vector of probabilities over the set of classes, i.e. if $\mathbf{1}$ is the all ones vector, then we require that $\mathbf{1}^\top \widehat{y}_i = 1$. This is enforced using softmax as the last layer.

Let t_i be the one hot encoded version of y_i, i.e.

$$t_{ic} = \begin{cases} 1, & \text{if } c = y_i \\ 0, & \text{otherwise} \end{cases}$$

.

Further let θ be the model parameters. We are now ready to write down our losses.

$$loss_1(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{c} t_{ic} \log(y_{ic}) + \lambda \|\theta\|^2$$

$$loss_2(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{c} t_{ic} \log(y_{ic}) + \mu \frac{1}{m} \sum_{i=1}^{m} \left| \sum_{c} c\widehat{y}_{ic} - y_i \right| + \lambda \|\theta\|^2$$

Note that $\mu = 0$, $loss_2$ is equivalent to the $loss_1$.

3 Model

3.1 Architectures

For our initial experiments, we used the AlexNet architecture [4] for our initial experiments. AlexNet has a smaller network with only 5 convolutional layers and 3 fully-connected layers. VGG has a larger network with 16 layers with several convolution layers and 3 fully connnected layers at the end. For both AlexNet and VGG architectures, we remove the last fc8 layer to produce 109 outputs, followed by a softmax.

3.2 Image Preprocessing

For AlexNet, we simply warped the images to size 227×227.

VGG expects input images to be of size 224 x 224. The network is trained on multiple crops of this size rather than simply resizing the training image to fit the network input size [8]. Since the images in our dataset have are nearly square in shape with the region of interest in the center, we crop narrow strips from both sides of the image to make it square (171 $\times$ 171). The image is then resized to 224 $\times$ 224 as desired by VGG. We also perform channel wise mean subtraction, identical to that done during the training of VGG on ImageNet.

3.3 Training

For AlexNet architecture, we set freeze the parameters in all the conv layers. We only train the fully connected layers. Training algorithm was Stochastic Gradient Descent with momentum. Except for the learning rate, the rest of the hyper-parameters were set to be the same as in the original ImageNet configuration.

For VGG architecture, we trained all the model parameters. The last layer was initialized to random gaussian values as per default initialization in tensorflow. We used Adam optimizer with various learning rates. Other values were same as those used for the VGG architecture.

4 Experiments and Results

4.1 AlexNet in Caffe

We varied the magnitude of the learning rate, and changed its policy from step to "poly".

Figure 1 shows the accuracy and train/test loss. The accuracy achieved is not good compared to the original paper [2]. We tried to train AlexNet using smaller learning rates, but we quickly ran into over-fitting problem as evident from the test loss in Fig. 2 and Fig. 3. As a result, we decided to switch to VGG network.

4.2 VGGNet in Tensorflow

We used Adam as the optimizer with a learning rate decay factor of 0.1 and a decay after every looking at 10000 images. We compared different learning rates, with and without regularization. Where used, the multiplicative factor for regularization was set to 5x10^{-4} [8]. We split the data randomly into a train set comprising of 80% of the total data and a validation set using the remaining 20%. We report accuracy and L1 error on the training and validation sets. The results can be seen in table 1.

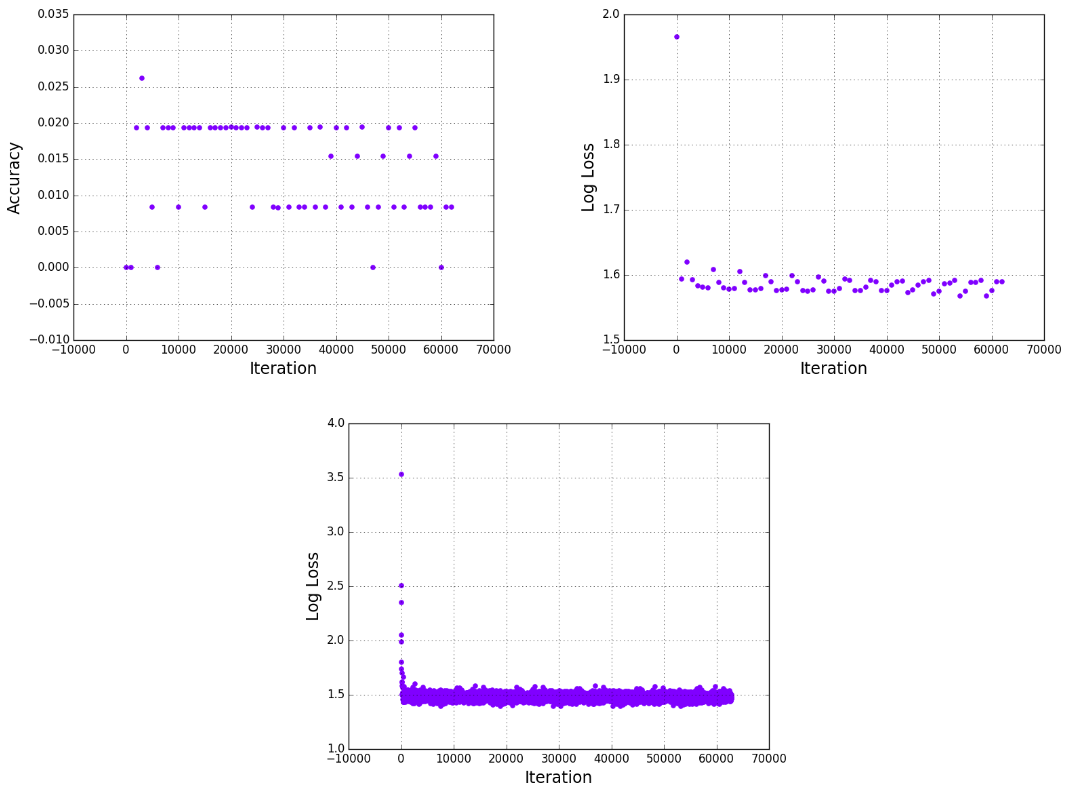


Figure 1: Test accuracy (left), test loss (right), and training loss (bottom) is shown for yearbook photo dating project when AlexNet is used. The learning rate hyper-parameter was set to $l_r = 0.01$.

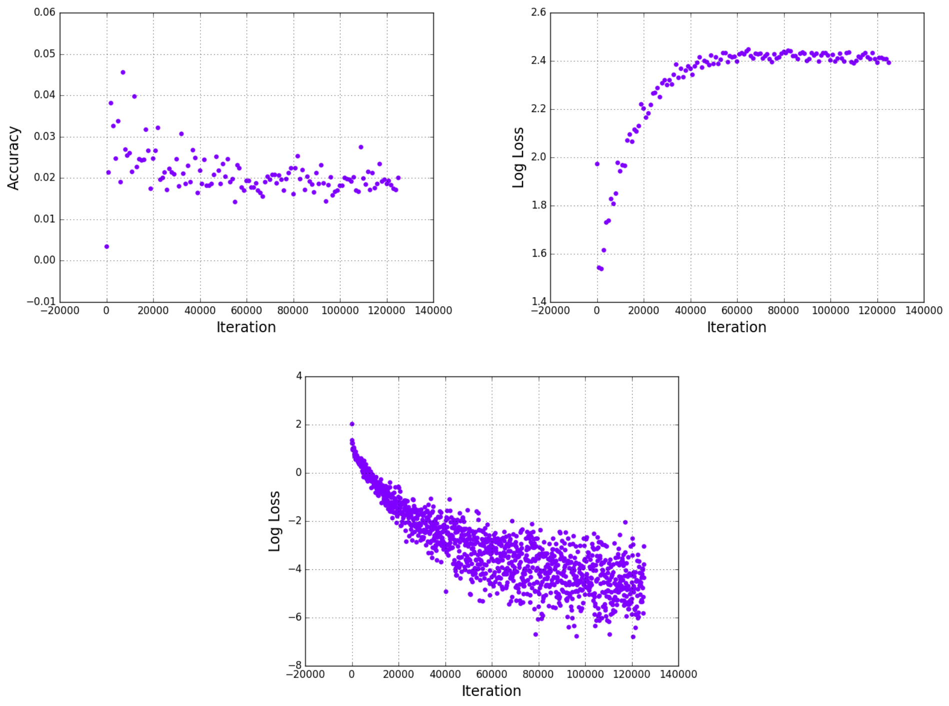


Figure 2: Test accuracy (left), test loss (right), and training loss (bottom) is shown for yearbook photo dating project when AlexNet is used. The learning rate hyper-parameter was set to $l_r = 0.001$. Overfitting is a problem that appears in the early stages of the training.

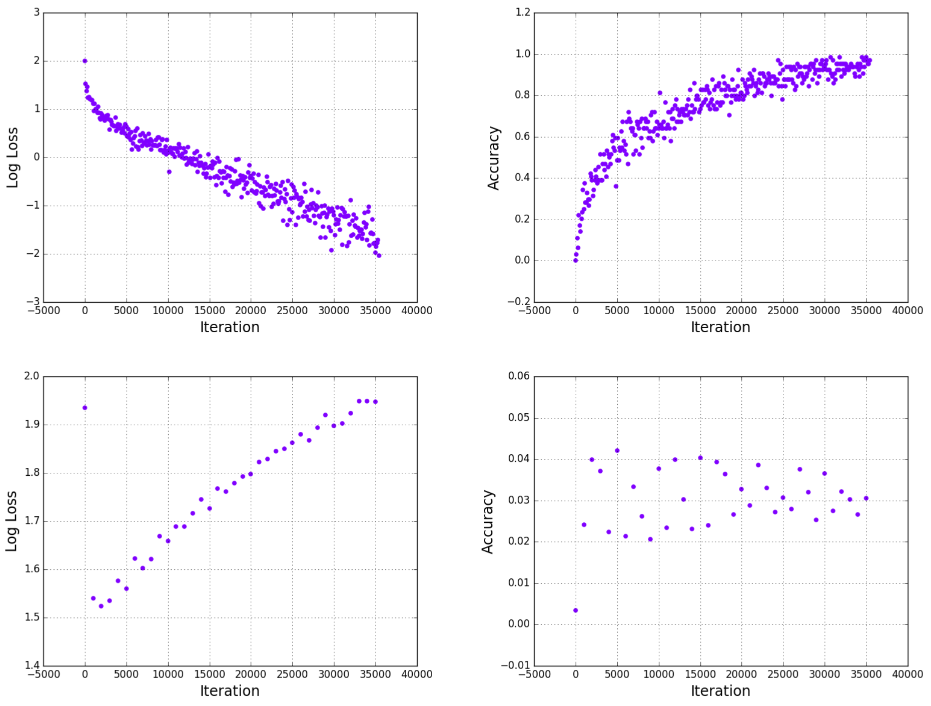


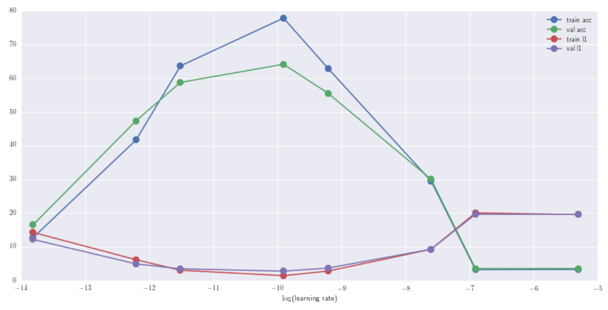
Figure 3: Train loss (top left), train accuracy (top right), test loss (bottom left), and test accuracy (bottom right) is shown for yearbook photo dating project when AlexNet is used. The learning rate hyper-parameter was set to $l_r = 0.0001$. Similar to Fig. 2, overfitting is a problem that appears in the early stages of the training.

Learning Rate	Regularization	Train Accuracy(%)	Train L1	Val Accuracy(%)	Val L1
0.005	No	3.30	19.59	3.57	19.66
0.001	No	3.24	20.07	3.57	19.66
0.0005	No	29.53	9.27	30.03	9.28
0.0001	No	62.97	2.81	55.6	3.70
0.00005	No	77.86	1.46	64.16	2.80
0.00001	No	63.67	3.05	58.79	3.49
0.000005	No	41.70	6.18	47.39	4.94
0.000001	No	12.68	14.29	16.51	12.21
0.005	Yes	3.29	19.61	3.57	19.66
0.001	Yes	3.24	20.49	3.57	19.66
0.0005	Yes	28.64	9.22	30.36	9.03
0.0001	Yes	61.08	3.15	53.55	4.13
0.00005	Yes	82.27	1.13	67.59	2.36
0.00001	Yes	71.93	2.29	63.43	3.01
0.000005	Yes	49.44	5.02	52.42	4.07
0.000001	Yes	13.05	14.3	15.49	12.19

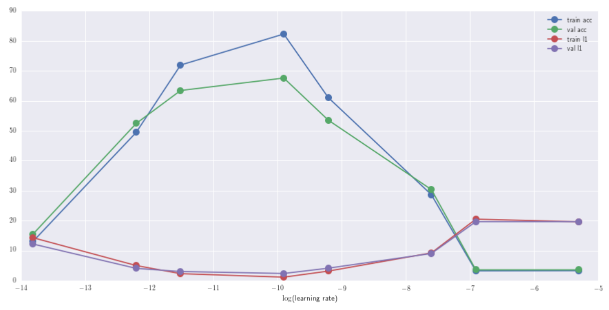
Table 1: Experiments on VGGNet with different learning rates with and without regularization

Now, we visually present the results shown in Table 1. We plot accuracy in (%) as well as the L1 error on the same plot to make it clear that both achieve their optimum value at the same learning rate value. Learning rate is on x-axis with log scale for better visualization. Results for without regularization are shown in Fig. 4a and those for with regularization in Fig. 4b.

It can be clearly seen that the best values for accuracy and L1 loss are obtained at the same learning rate setting.



(a) VGG without regularization



(b) VGG with regularization

Figure 4: Caption place holder

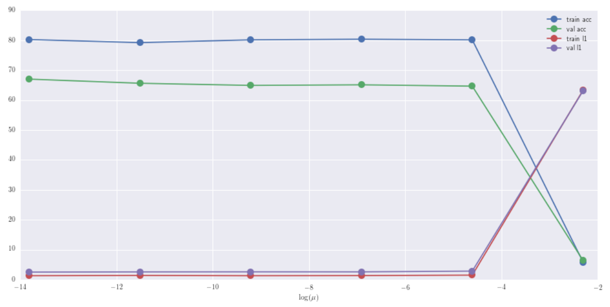


Figure 5: VGG combined loss

For some of the best performing models we see that train metrics are better than test metrics. This is usually an indication of overfitting, but in our case, they are monotone w.r.t each other. Thus, if one is to choose the model based on validation accuracy, we would still pick the one that was best on the training set. For the best setting, we achieve best training accuracy of 82.27%, validation accuracy of 67.59%, training L1 loss of 1.13, and validation L1 loss of 2.36. This is significantly better than Ginosar *et al.* [2], which achieves only 11% accuracy. They do not report the mean L1 loss, but the median L1 loss, which is probably smaller than mean L1 is still at 4, whereas we achieve mean L1 of 2.36 on test set.

Next we present results of experiments on VGGNet using the combined classification and regression loss, i.e. $loss_2$. We used the same hyperparameters as in section 4.2. They are also reported on the same train-validation split. We use L2 regularization with $\lambda = 5\text{x}10^{-4}$ and a learning rate of 0.00005 as this model produced the best results on the validation set. We compare the performance using different values of the parameter μ. The results are presented in table 2. These results are also shown visually in Fig. 5. Note that we plot μ on the x-axis on log scale for visual clarity. As a consequence, datapoint with $\mu = 0$ is not plotted. Clearly, models with regression loss were very close, but did not do as well as the model without L1 regression term.

μ	Train Accuracy	Train L1	Val Accuracy	Val L1
0.0	82.27	1.13	67.59	2.36
0.000001	80.24	1.30	67.01	2.51
0.00001	79.21	1.40	65.63	2.58
0.0001	80.14	1.30	64.91	2.60
0.001	80.35	1.35	65.11	2.59
0.01	80.13	1.50	64.67	2.85
0.1	5.86	63.32	6.39	63.11

Table 2: Experiments on VGGNet using combined classification and regression loss

5 Visualizations

To better understand what the model has learned, we also did a few visualizations. We follow the two methods given in Simonyan *et al.* [7]. These are briefly described here.

5.1 Image Class saliency map

Given an image I and a class c, we try to find which pixels in the image were most responsible for making the model think that the image might be from that class. This is known as Image-Class Saliency Map.

One way to do this is to compute the gradients of the image with respect to the score for that class. Let $S_c(I)$ be the logit for the class c when image I is given to the trained model. Then the magnitude of gradient of S_c with respect to I is a locally linear approximation to the Image-Class saliency map.

5.2 Class Maximal Image

To visualize the concept of a class as learned by the model, we try to find the image that maximizes the score for that class. Notice that the image can simply be scaled to improve the scores, and thus we also want to regularize the image. Thus, as before for a class c, let $S_c(I)$ be the score for class c with image I. So we want

$$I^* = \arg\max_I \{S_c(I) - \lambda \|I\|^2\}$$

, where λ is a hyperparameter.

To approximately solve this problem, we can start with a uniform I_0 and do gradient descent with respect to I.

5.3 Results

We compute these visualizations only for the best model we found.

In 6, we show some sample image class saliency maps. It can be seen that the model focuses on the forehead, cheek and chin areas. Eyes are mostly ignored. For the first image nose is an important cue.

Next, we show class maximal image for a few classes in 7. These have a few patterns, and if you squint a little and imagine the rest, you can see eyes and noses. We conclude that these visualizations are not very informative.

6 Conclusion

In this report we present our attempt to solve the task of predicting the years a photo was taken from the year-book photos. We used Convolutional Neural Nets as our classifier. We explored several architectures and loss types. With careful tuning of parameters, we have shown substantial improvement over the best published result. Finally, we also compute some visualizations to gain some insights into what the model has learned.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

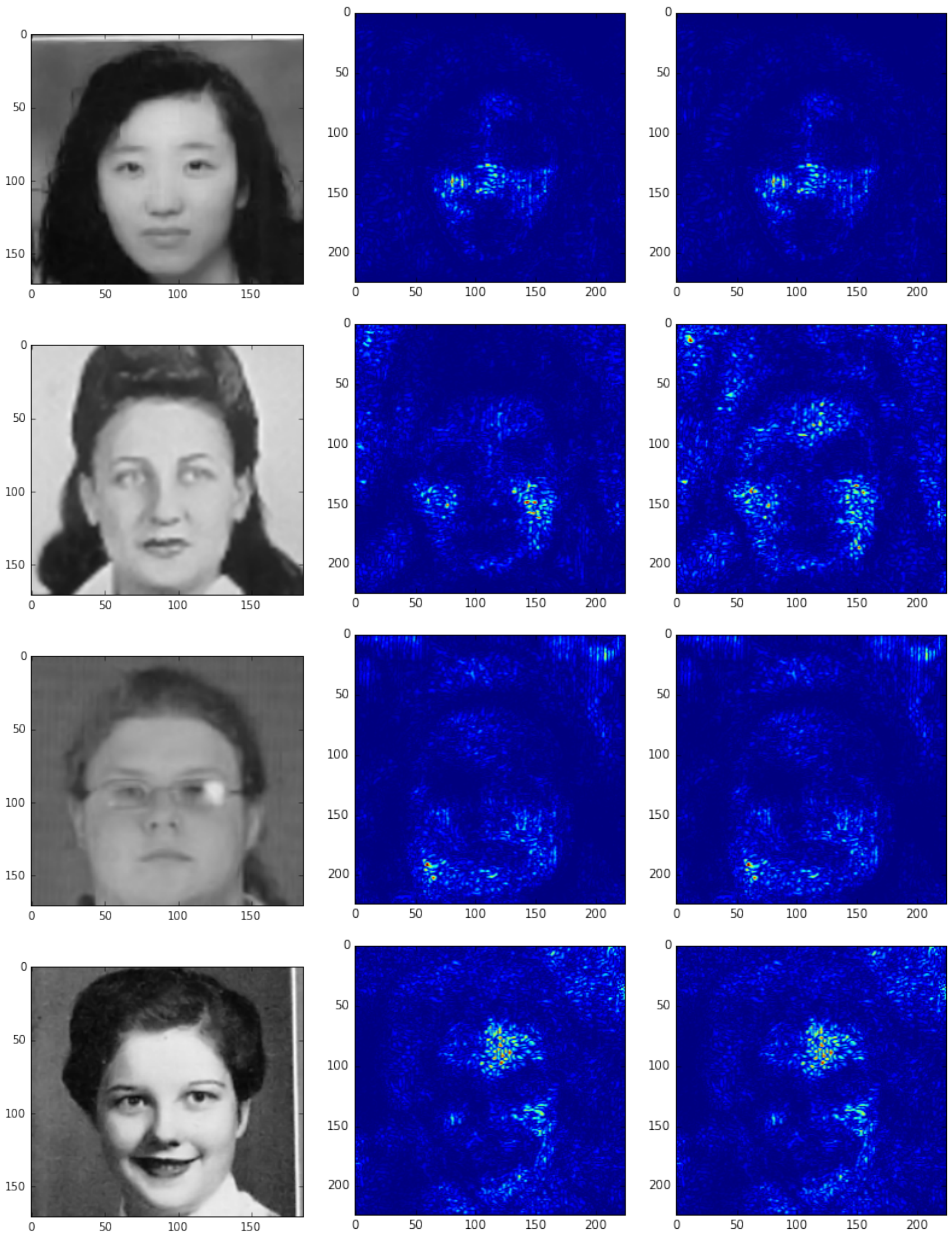


Figure 6: Input image (left), gradient wrt predicted class (center) and gradient wrt true class(right).

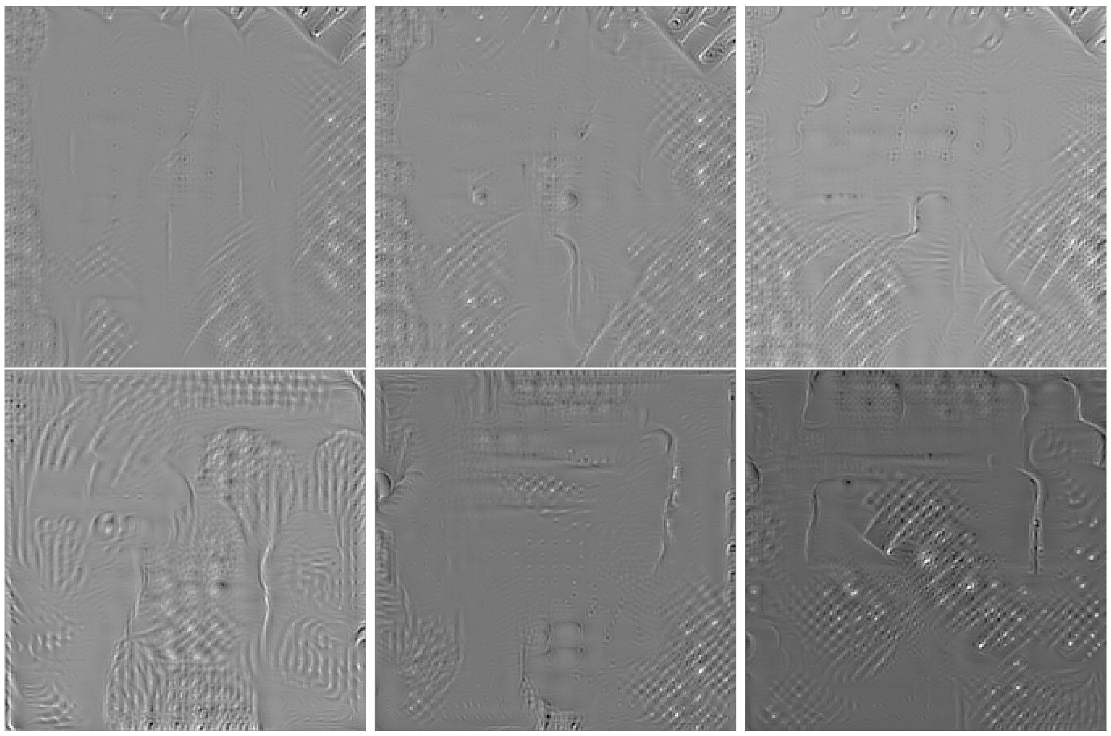


Figure 7: Class maximal images

[2] Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, and Alexei A. Efros. A century of portraits: A visual historical record of american high school yearbooks. In *International Conference on Computer Vision*, 2015.

[3] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, volume 1, page 6, 2015.

[6] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA, 2014. IEEE Computer Society.

[7] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations Workshop*, 2014.

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

[9] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4534–4542, 2015.

[10] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5, 2015.