A Survey on Facial Feature Detection Methods and Their Application for Automatic Lip Reading

Calvin Chang Krish Kabra Amit Mondal
Satvik Anand
{calvinchang33, krish97, amitmondal, satvik29} @g.ucla.edu
University of California, Los Angeles
Los Angeles, CA 90095

Abstract

Facial feature detection (FFD) is an important task used in a multitude of applications including facial recognition, animation, expression analysis, and 3D modelling. In this report, we survey 6 well-established FFD methods: active shape models, active appearance models, active contour models (ACM), localized region-based ACM, supervised descent fitters, and ensemble of regression trees. We utilize the LFPW and 300-W IBUG facial databases to train our models, as well as to qualitatively and quantitatively analyze model performance. Finally, we show an example application of FFD to the automatic lip reading task using two deep learning frameworks: a feature-only network and an end-to-end network.

Contents

1. Introduction 1

2. Facial Feature Detection Methods 2
2.1. Active Shape Models . . . 3
2.1.1 Implementation . . . 3
2.1.2 Performance . . . 3
2.1.3 Examples . . . 3
2.2. Active Appearance Models . . . 4
2.2.1 Implementation . . . 4
2.2.2 Performance . . . 4
2.2.3 Examples . . . 4
2.3. Active Contour Models . . . 5
2.3.1 Implementation . . . 5
2.3.2 Performance . . . 5
2.3.3 Examples . . . 5
2.4. Localized Region-Based Active Contour Models . . . 5
2.4.1 Implementation . . . 5
2.4.2 Performance . . . 6
2.4.3 Examples . . . 6
2.5. Supervised Descent Method . . . 6
2.5.1 Implementation . . . 6
2.5.2 Performance . . . 7
2.5.3 Examples . . . 7
2.6. Ensemble of Regression Trees (Dlib) . . . 8
2.6.1 Implementation . . . 8
2.6.2 Performance . . . 9
2.6.3 Examples . . . 9
2.7. Discussion . . . 10

3. Application for Automatic Lip Reading 10
3.1. Feature-based approach . . . 10
3.2. End-to-end approach . . . 10
3.3. Results . . . 11

4. Conclusion 11

1. Introduction

Automatic facial recognition and analysis is a core topic in computer vision. Crucial to this field is the important task of facial feature detection (FFD), which is the process of finding semantic features on a given 2D image of a face. These features typically consist important facial components such as the eyes, mouth, nose and chin. Detecting and tracking these features over time enables applications including head pose estimation [1], facial expression analysis [2], facial recognition [3], facial animation [4], 3D facial modelling [5] and visual speech recognition [6, 7].

Typically, FFD consists of learning the location of facial fiducial points through supervised or semi-supervised training on manually labelled images containing the fixed feature point annotations. Depending on various applications, different numbers of facial feature points are labeled as, for example, a 21-point model [8], 29-point model [9] or 68-point model [10]. Such point detection methods are distinct from the feature detection methods used in image registration, image stitching, and object recognition applications [11, 12, 13, 14, 15], where general key points are identified using image-specific features such as edges, corners, blobs and ridges. Alternatively to point-based method, FFD can compromise of learning bounding boxes [16] and then using segmentation methods to more accurately extract the facial component [17, 18, 19].

Facial feature point detection (FFPD), also known as facial landmark or facial fiducial point detection, can be separated into 2 categories: generative and discriminative methods. Generative methods build an object template, and then iteratively deform the template to the input image. Such methods include active shape models (ASM) [20], active appearance models (AAM) [21], and constrained local models (CLM) [22]. Discriminative methods learn to separate the object from the background through a binary classification of the image pixels. Among such methods, cascaded regression models (CRM) have gained widespread popularity due to its excellent performance and low complexity [23, 24, 25]. Deep learning approaches have also been proposed with the rise in publicly available labelled databases [26, 27].

FFD using bounding boxes and segmentation is not as common as FFPD. Nevertheless, previous works have typically used variants of active contour models (ACM) [28], also known as snakes, to segment facial features [17, 18, 29]. Bounding boxes are used to initialize the snakes with simple shapes such as ellipsoids for the mouth and eyes, and triangles for the nose.

In this report, we survey and comparatively analyze the following FFD methods:

- Active Shape Model (ASM) [20]
- Active Appearance Model (AAM) [21]
- Active Contour Models (ACM) [28]
- Localized Region-Based ACM (LACM) [30]
- Supervised Descent Method (SDM) [23]
- Ensemble of Regression Trees (ERT) [24]

In order to train and evaluate the FFD methods, we utilize the LFPW and 300-W IBUG facial point annotations databases [10, 31, 32]. The ground truth facial point annotations follow the Multi-PIE 68-points mark-up [33], and were created by using a semi-automatic methodology [31] followed by additional manual correction. The LFPW database consists of 1,432 faces [9], whereas the 300-W database consists of 300 Indoor and 300 Outdoor in-the-wild images. Both databases downloaded images from the web using simple text queries on sites such as google.com, flickr.com, and yahoo.com.

Finally, to show an example application of FFD, we implement two deep learning frameworks for the task of automatic lip reading (ALR) using tracked visual facial features. ALR is extremely challenging— for context, human level accuracy is atrociously poor with lip reading performance metrics of $17 \pm 11\%$ for monosyllabic words and $21 \pm 12\%$ for compound words, from word banks of 30 [34]. One reason for this is because of human inability to visually extract the difference in consonant phonemes like "p" and "b" [35]. More over, in the absence of context, discerning and differentiating the difference in lip, teeth and tongue movements contributes to the resulting poor human performance [30]. Without a doubt, there is a need for an automated process for lip reading. ALR performed by machines will impact applications such as hearing aids, silent dictation in public spaces, security, speech recognition in noisy environments, biometric identification, and silent-movie processing [36].

ALR is approached by various flavors, but most typically follows a common suit of first learning visual features and then performing speech prediction. The first ALR systems mainly consist of image transforms or appearance-based features combined with hidden markov models that use short context information to model the temporal dynamics of the speech sequences [37]. However, these systems could only tackle simple recognition tasks such as alphabet or digit recognition. The recent rise in deep learning (DL) architectures in combination with the availability of large-scale databases has enabled systems to undertake more complex and realistic settings leading to systems that target continuous lip-reading [7].

For this report, we follow suit of using DL to perform ALR on single word utterances. We use two frameworks: a feature-only based network and an end-to-end network. The feature-only network serves as a simple proof-of-concept that FFD can directly be used for ALR. We feed the sequence of facial feature coordinates tracked during the video utterance directly into a bi-directional recurrent neural network (RNN) [38]. The end-to-end network used is based on the well-established LipNet architecture [36] that feeds cropped video frames of the mouth obtained using FFD. The network is modified to predict words as opposed to original sentence-level task. We utilize 20 word utterances from the lip reading in the wild (LRW) video database for model training and evaluation [39]. Due to time and computational resource limitations, we only use facial features tracked using the ERT method due to its high accuracy and speed. We show both models can adequately learn to classify the word utterances.

2. Facial Feature Detection Methods

In this section, we qualitatively and quantitatively compare the various facial feature tracking methods we evaluated in this project. For each method, we will include a brief overview of how it works, describe secondary characteristics of the model (runtime performance, amount of data needed to train), show examples of facial features extracted, and show accuracy measurements.

To calculate accuracy for our methods, we use a standard

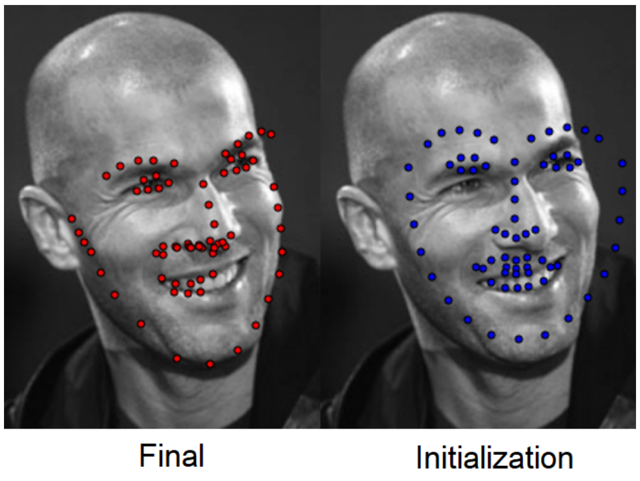

Figure 1. Example of poor ASM performance. Error from variance in depth and non-uniform lighting. Blue (right) is initialization and red (left) is final fit.

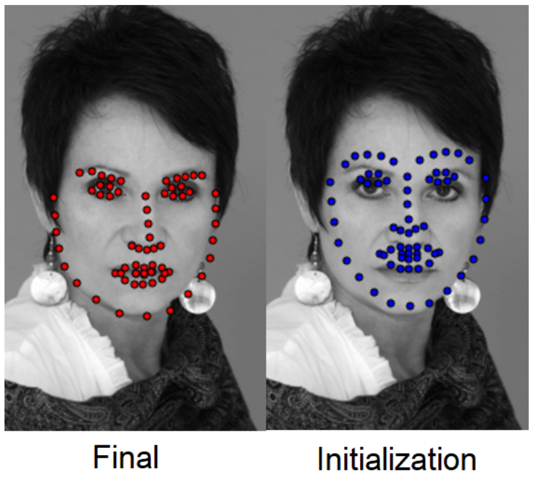

Figure 2. Example of good ASM performance. Low error due to features being in the same depth plane and uniform lighting. Blue (right) is initialization and red (left) is final fit.

Euclidean distance measure:

$$\mathcal{F}(s, s*) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(s_{i,x} - s^*_{i,x})^2 - (s_{i,y} - s^*_{i,y})^2} \quad (1)$$

where s and s^* are the final and ground truth shapes, respectively. $(s_{i,x}, s_{i,y})$ represents the ith shape in the final image, and $(s^*_{i,x}, s^*_{i,y})$ represents the ith shape in the ground truth image. Ideally, we could normalize this value by dividing it by some quantity proportional to the size of the ground truth's bounding box, since without this normalization, larger images will inherently have a larger error simply because points are further apart. However, because some of the models used in this project don't support this normalized measure, we simply use standard Euclidean distance. Fortunately, because we are testing all the models on the same dataset, LFPW, this shouldn't cause any issues for comparison purposes. To generate an accuracy score, we calculate the above score for every image in the test set of LFPW, and then take the max, min, and mean of these accuracy scores.

2.1. Active Shape Models

For facial feature tracking, ASM [20] is able to capture fine detail and generalizes to a larger variation of shapes compared to snakes. ASM in ALR, has shown relatively positive results as an exhaustive approach for limited variation and small datasets. Fundamentally, ASM relies on a priori knowledge about the deformation statistics of a labeled training set. In addition, ASM receives a performance boost if the environment can be correctly predicted, modeled or constrained.

2.1.1 Implementation

The ASM implementation was done using a CLM in an attempt to account and handle more variation [22]. CLM achieves this by not only considering the shape model, but also considering the response of each feature formulated as patches about the features. Each patch constitutes an image response, which are represented by holistic features based on a gradient and are then fed into PCA to form a linear model to learn from during training. During fitting time, CLM performs a similar iterative approach as ASM in which the model tries to maximize an objective function consisting of the image response and the log likelihood given the shape parameters and eigenvalues [22].

2.1.2 Performance

This improved ASM model trains slow compared to the rest of the surveyed models in this paper. Training on the LFPW dataset consisting of 811 images, takes about 1 minute per scale on an 8 core AMD 3700x. On average, it takes about 0.41 seconds to perform feature extraction from a single image. Making this ASM method not viable to be tested in our feature-only network given the time constraint.

In terms of accuracy for ASM, the minimum is 2.23, the maximum is 1528, and the mean is 38. Note that the accuracy scores are created from an initialization based on Dlib's face detector.

2.1.3 Examples

Figure 1 is an example of how the improved ASM model struggles to correctly fit images that have features lying in different depth planes. In addition, the model tends to struggle with complex scene lighting. This is most likely due to

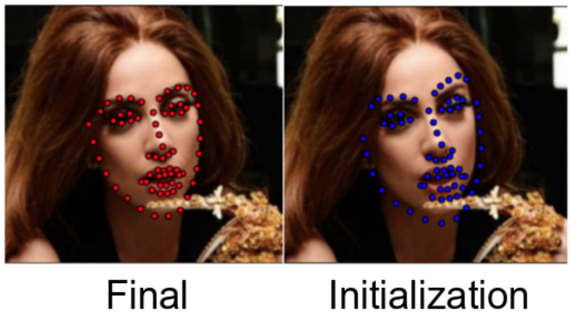

Figure 3. AAM produces good results when given a close initialization.

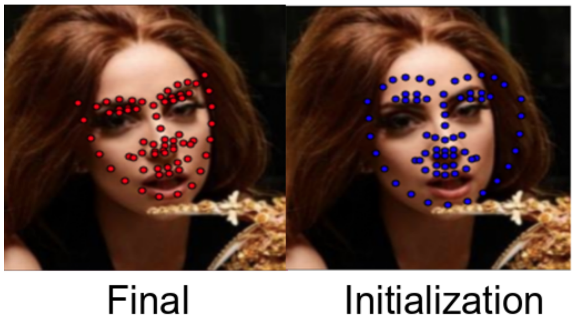

Figure 4. AAM can get tricked by misleading features when given a bad initialization.

the fact LFPW lacks enough data given a particular change in depth or lighting. In addition, the holistic features are captured through a gradient, which can be heavily influenced by changes in lighting intensities. One thing to note, is that this method sometimes does a good job despite poor initialization as seen in Figure 1. However, the model cannot finish the job since it is not invariant to changes in depth or lighting. Figure 2 results in a nearly optimal fit, because the scene has low variance in feature depth and the lighting is uniform throughout the scene.

2.2. Active Appearance Models

Although ASMs are relatively fast thanks to learning eigenfaces through PCA, they are not as robust when new images are introduced. As a result, ASMs may not converge to a good solution, especially given the wide variation in human faces. AAMs [21] fix this by incorporating gray-level information with the shape information into a single statistical model. The result is a model with similar benefits to ASMs, such as their speed, while also having better representational power thanks to the addition of texture information, which allows them to be more robust overall.

2.2.1 Implementation

The implementation of AAM used in this project uses a holistic appearance representation obtained by warping the texture into the reference frame with a non-linear warp function. In this case, the reference frame is the mask of the mean shape's convex hull. The warp function is a piecewise affine warp. The PCA appearance model has 1034 components, and the PDM (point distribution model) shape model has 132 components. We use the Lucas-Kanade gradient descent image alignment algorithm for fitting the appearance model to images.

2.2.2 Performance

As expected, the model was very quick to train, completing the LFPW dataset in under 5 seconds on a 6-core Intel i7-8750H. In addition, the model ran relatively quickly when extracting images. On average, our AAM model took .08 seconds to extract features from a typical 500x800 image from LFPW.

In terms of accuracy for AAM, the minimum is 2.67, the maximum is 8294, and the mean is 427. Note that the accuracy scores are created from an initialization based on Dlib's face detector.

2.2.3 Examples

Figure 3 shows a good initialization in blue, and the final fitting result in red. As we can see, with a good initialization, our AAM performs very well, perfectly fitting the eyebrows, nose, and lips. The eyes and chin are slightly off due to shadows, but subjectively, the fitting result is perfectly acceptable for our purposes.

Figure 4 shows a bad initialization in blue, initialized via our facial detector, and the final fitting result in red. We can see that if the initialization is shifted significantly, our AAM detector can easily get "tricked." In this case, we see the

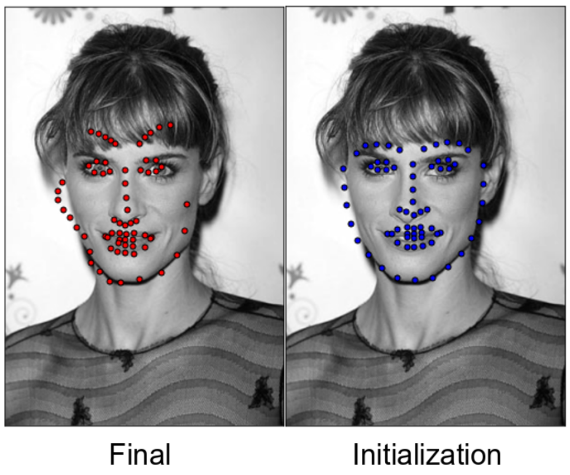

Figure 5. Even some fairly close initializations can produce bad results. In this case, the AAM model is tricked by the texture in the hair and the shadow behind the face.

AAM fitting the eye features to the subject's eyebrows. In addition, because the shadow at the bottom of the face happens to line up with the subject's lip, the bottom of the face mesh is aligned with the shadow and the mouth. Finally, the mouth portion of the mesh is aligned with a shadow between the mouth and nose. We can see that for our purposes, this AAM is not robust enough for use with our Dlib face detector.

We provide one more example in Figure 5 to show that even with a fairly close initialization, the AAM is still not robust enough for our purposes. With this last initialization in blue, we see that the boundaries of the face mesh almost perfectly align with the subject's actual face, and the eyes, nose, and mouth are very close to their ideal final position. Unfortunately the AAM seems to get confused by shadows once again, and completely misaligns the eyebrows due to the hair texture, and misses the left edge of the face due to a shadow in the background.

2.3. Active Contour Models

The conventional ACM [28] algorithm contours objects by making an initial closed curve deform via minimizing a regional energy. Using ACM to extract lip contours for ALR presents some challenges because of its performance with uneven lighting and parameter initialization.

2.3.1 Implementation

The implementation of ACM used in this project first finds edges on the image using a Sobel filter and then superimposes intensity and edge images. Then it interpolates the image for smoothness and builds the snake shape matrix for Euler equations by explicitly time-stepping and minimizing energy. One can set the parameters for contour length, smoothness, attraction to brightness and darkness, distance moved each iteration, boundary conditions and convergence criteria.

2.3.2 Performance

The challenge of using ACM to detect the lip contours in images is that one needs user input to initialize the parameters of the active contour, which is unfeasible for training on large datasets. With improper initialization the ACM tends to get confused by uneven lighting and deformities around the lip. It is challenging to select the right parameters that cause the ACM to contour the lips. The ACM model took between 1 and 2 seconds to extract the lip contour for most of the images in the LFPW dataset with some outliers taking about 3-4 seconds. The time taken is affected by the initialization parameters.

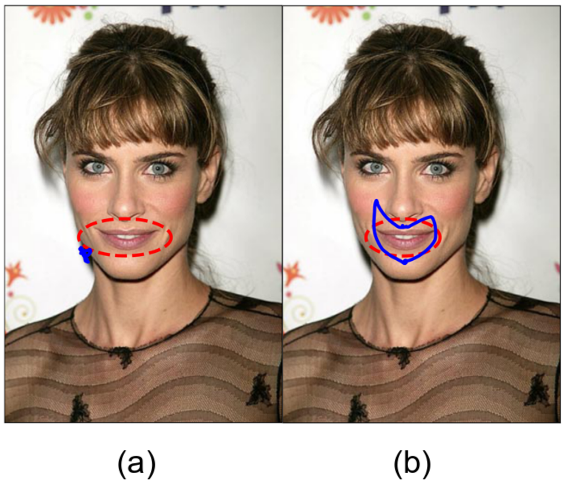

Figure 6. The red dotted line in these figures represents the initialization and the blue represents the final result. (b) has a better result than (a) because of better initialization but both are still very inaccurate.

2.3.3 Examples

In Figure 6a ACM's initialization was very poor and we weren't able to get a good fit at all. The model got confused by a shadow on the side of the person's face. To counter this, in Figure 6b, we changed the parameter that controls attraction to darkness to avoid darkness and it seems to have found a better fit but still gets confused by uneven lighting on the face. We can observe that conventional ACM is not robust enough for the ALR problem.

2.4. Localized Region-Based Active Contour Models

When images have heterogeneous or complex components, LACM [30] are known to perform better than ACM because they are able to separate the image into internal and external components and ignore the information from the external components in finding edges. Edge-based active contours utilize image gradients to find object boundaries. However LACM can be sensitive to noise and is, like ACM, highly dependent on initial curve placement and is also susceptible to uneven lighting and deformities around the lips.

2.4.1 Implementation

The implementation of LACM is based on level sets that are evolved iteratively to minimize an energy, which is defined by weighted values corresponding to the sum of differences in intensity from the average value outside the segmented region and inside the segmented region and a term which is dependent on the length of the boundary of the segmented region. At each iteration you calculate new level set values

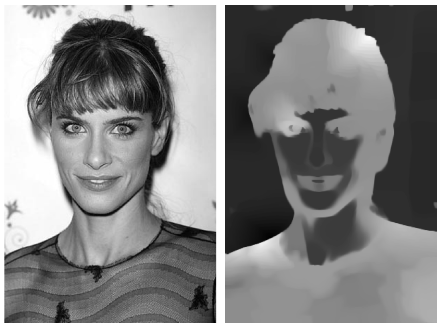

Figure 7. The original picture compared with the final level set after applying LACM algorithm. Lip intensity is clearly differentiated.

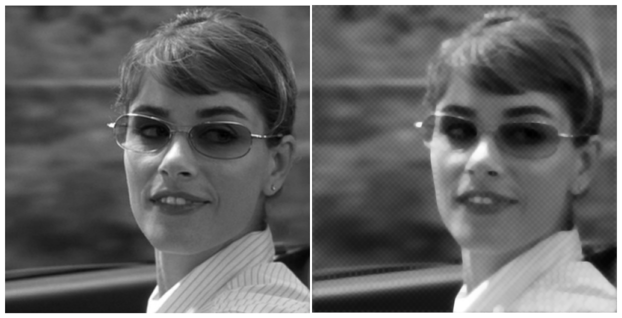

Figure 8. The original picture compared with the final level set after applying LACM algorithm. Lip intensity is almost identical to original image.

and compare it to the previous level set to see if minimization should continue.

2.4.2 Performance

LACM is also dependent on the user initialization to get a good approximation but performs better than ACM in most cases. However, it can still get confused by uneven lighting and lip deformities. The LACM implementation is computationally slower than ACM and takes between 5-8 sections to calculate the final level set for all images in the LFPW dataset.

2.4.3 Examples

In Figure 7, we can see what happens when the algorithm manages to segment the different parts of the face effectively. The lips are clearly contoured a different intensity than the area surrounding the lips.

In Figure 8, basically no segmentation has taken place. The final level set is considerably similar to the original image. This is likely due to somewhat homogeneous intensities all across the image which leads to a mostly unchanged level set.

2.5. Supervised Descent Method

SDM is a more recent method for extracting facial features, introduced in 2013 [23]. Many computer vision problems, including facial alignment, can be viewed as solving some nonlinear optimization problem. For these types of problems, Newton's method is often chosen because of its quadratic convergence rate and its guarantee to converge. Newton's method has the following update step: $\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}^{-1}(\mathbf{x}_k)\mathbf{J}_f(\mathbf{x}_k)$, where $\mathbf{H}(\mathbf{x}_k)$ and $\mathbf{J}_f(\mathbf{x}_k)$ are the Hessian and Jacobian evaluated at $\mathbf{x}_k$, respectively. However, for computer vision problems, there are a few challenges with this method. The Hessian might be positive definite at the minimum, but not elsewhere, so Newton method's steps might not be taken in the descent direction. In addition, Newton's method requires the function to be twice differentiable. This can be expensive to estimate, for example in the case of SIFT features, which are non-differentiable [12]. Finally, the Hessian matrix can be extremely large, and inverting it can be even more expensive.

To solve these issues, SDM takes a different approach to optimizing some nonlinear-least squares (NLS) function. During training, SDM is given a set of functions $f(\mathbf{x}, \mathbf{y}^i)$, where $f(\mathbf{x})$ is a non-linear function of image features such as SIFT, and $\mathbf{y}^i$ represents different locations (people). For each of these examples, the minima is known, and SDM learns a series of parameter updates which incrementally minimizes the mean of all NLS functions during training. Each update consists of some sample-specific component (e.g. $\mathbf{y}^i$), and some generic descent directions $\mathbf{R}_k$. During fitting, given some unseen $\mathbf{y}$, we can generate this update by projecting $\mathbf{y}$-specific components onto the learned descent direction $\mathbf{R}_k$. The NLS used for facial feature detection is $f(\mathbf{x}_0 + \Delta x) = \|\mathbf{h}(\mathbf{d}(\mathbf{x}_0 + \Delta x)) - \phi_*\|_2^2$, where $\phi_* = \mathbf{h}(\mathbf{d}(\mathbf{x}_0))$ represents the SIFT feature values in the manually labelled landmarks. ϕ_* and $\Delta\mathbf{x}$ are known in the training images. A primary difference between SDM and other models, such as AAM, is that AAMs only use one-step regression, which have been shown to lower performance.

2.5.1 Implementation

We used several implementations for this project. The first version uses the non-parametric Newton algorithm and incremental regularized linear regression (IRL regression). We use 30 perturbations per shape, and only two scales (0.5 and 1.0). For the first version, we use a 17x17 patch extracted from the image, and don't use any custom image features. As we'll show in the next section, this first version, train solely on LFPW, did not perform particularly well.

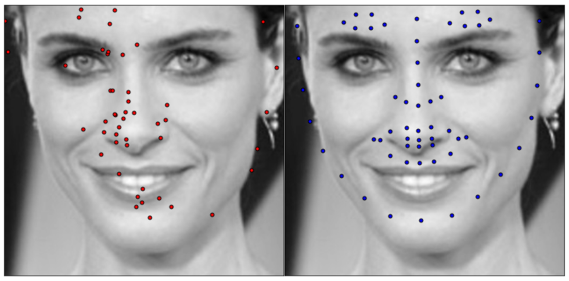

Figure 9. Example using the first SDM model. Without sufficient training data and regularization, SDM model overfits and can produce completely garbled results.

The second version has several major differences. First, we use a regularized version of the SDM model, which we'll see is extremely important to prevent overfitting. Second, we use a standard 128-length SIFT feature vector as our patch feature. We use a 24x24 patch size, and we set the regularization parameter α to 10.

The third version makes several slight changes. First, we only use 3 perturbations per shape. Second, we use 4 scales this time, two at 0.5 and two at 1.0. The two at scale 0.5 use 32x32 and 24x24 patches each, and the two at scale 1.0 use 24x24 and 16x16 patches each. Finally, we now use a slightly modified Hellinger SIFT feature vector as our patch feature. Hellinger is a more robust distance measure for SIFT than standard Euclidean distance. As we will show, adding in more scales while also using SIFT features and training on a much larger data set leads to much improved performance.

2.5.2 Performance

SDM is notable because it took significantly longer to train than prior methods, such as AAM. On the same 6-core i7 CPU we used in the AAM training, it took our first version of SDM about 11 minutes to finish training on LFPW, which is several orders of magnitude longer than our AAM version. This makes sense, since SDM is doing significantly more processing per image, and the computations are no longer as simple as PCA. Keep in mind that this is the performance of our simpler version of SDM, which does not use SIFT, and only uses 2 scales. This version of SDM was extremely fast for actually performing feature extraction, taking about .01 seconds to extract features from the same typical 500x800 image from LFPW. We can see that SDM is very expensive to train, but has impressive performance for testing.

Our second version of SDM described above, with larger patches and SIFT features, took about half an hour to train on LFPW. We can see that adding in a custom feature descriptor for patches significantly increases training time.

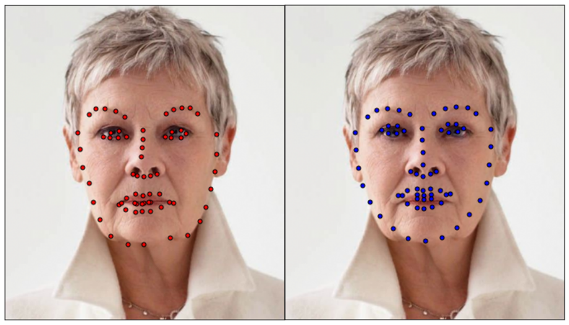

Figure 10. Good example using the second SDM model. With regularization, the model can produce good results as long as it is given a good initialization

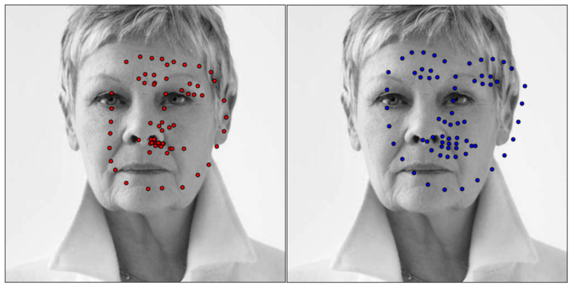

Figure 11. Poor example using the second SDM model. Even with regularization, the model is still not robust to bad initialization. More data and more scales are needed to improve model performance.

For our best-performing SDM model, the minimum is 1.36, the maximum is 403, and the mean is 19.9. Note that the accuracy scores are created from an initialization based on Dlib's face detector. We can see that this model performs significantly better than our AAM model, which had a mean error of 426, which is worse than our SDM model's maximum error.

2.5.3 Examples

In Figure 9, we can see an example of the first iteration of our SDM model. The initialization, seen in blue, is provided by Dlib's face detector. We can see it is shifted upwards slightly from the correct position, but is relatively close to the true features. However, in red, we can see that the final fitting result is basically a garbled mess. It is possible that a better initialization would have produced a better fitting result, but when we actually use this model we must accept whatever initialization that our face detector gives us. In this case, it is not clear what exactly the model is trying to fit. Because of the dramatic improvements that regularization give us, we can say with a fair amount of confidence that this initial model has overfit. In short, this initial SDM has

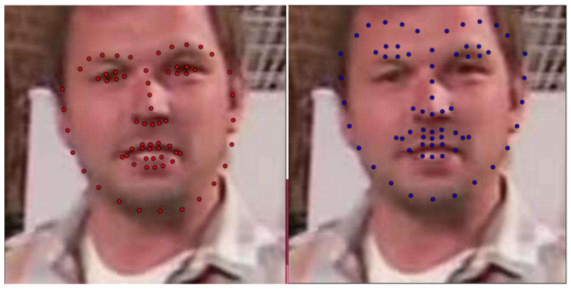

Figure 12. Example using the third SDM model. With regularization and sufficient training data, SDM has excellent performance.

unacceptable performance.

In Figure 10, we see our second SDM model is greatly improved. With a good initialization, it is able to produce an excellent fitting result that almost perfectly matches the true features. It is slightly thrown off by the sheen on the left cheek, but overall the fitting result is excellent. However, a bad initialization is enough to produce an unacceptable fitting result. In Figure 11, we use a shifted and tilted initialization for this fitting, and we can see that the final fitting result matches almost none of the actual features. The model is completely thrown off by inconsistent lighting and the bad initialization, and produces an unacceptable fitting result. Thus, with our second SDM model the performance has improved dramatically, It is now somewhat similar to our AAM model, where a good initialization can produce excellent fitting results but a bad initialization can "trick" the model into fitting to shadows or other misleading image features. Given that we have trained this model on the same amount of data, it is clear that the image patch features and the regularization are responsible for the improvement. This model is much better, and better than many of the prior appearance and shape models, but still needs to be improved before we can use it in our deep learning model.

Figure 12 shows an example from our third SDM model, which has 4 scales, is regularized, and uses the Hellinger SIFT feature vector. It is also trained on significantly more data than only LFPW. We can see that even when given a similar initialization to our first example, where the face mesh is offset vertically, we end up with a nearly perfect fitting result. It is clear that regularization, adjusted features, and significantly more data can make SDM a very accurate facial feature extraction model. This final SDM model has good enough performance for use in the deep learning model.

2.6. Ensemble of Regression Trees (Dlib)

The newest feature detection method we tried is ERT, introduced in 2014 [24]. Much like SDM, ERT is a cascaded regression method that attempts to improve over prior facial feature detection methods in some novel way. There are several important issues in feature extraction that ERT builds on prior methods to solve.

The first is that the extracted facial features in the vector representation of a face image can vary greatly due to different facial lighting conditions. This makes estimating the shape from these features difficult, and conversely, we need an accurate estimation of the shape to extract accurate features. Rather than regressing shape parameters based on features extracted in the global coordinate system of the image, the image is transformed to some normalized coordinate system based on the current shape estimate, and then features are extracted to produce an update vector for the shape parameters. This iterative update procedure (similar to an EM algorithm) is repeated until convergence.

The second issue is that estimating the shape (a high-dimensional vector) is a non-linear problem with many local minima. To reduce the number of shapes considered during inference and avoid local optima, we assume that the estimated shape must lie in some linear subspace. This subspace can be discovered by finding the principal components of the training shapes, for example. The authors exploit the fact that a certain class of regressors are guaranteed to produce predictions that lie in a linear subspace defined by training shapes, so there is no need for additional constraints.

Each regressor in the cascade is learned via gradient boosting with a squared-error loss function. We let $\mathbf{S} = (\mathbf{x}_1^T, \mathbf{x}_2^T, ...\mathbf{x}_P^T)$ be the coordinates of our facial landmarks in our image I. Our current estimate of $\mathbf{S}$ is $\hat{\mathbf{S}}^{(t)}$. We compute an update for the estimate like so: $\hat{\mathbf{S}}^{(t+1)} = \hat{\mathbf{S}}^{(t)} + r_t(I, \hat{\mathbf{S}}^{(t)})$, where $r_t(.)$ is one regressor in the cascade. Each regressor makes its predictions based on features from the image, I, indexed relative to the current estimate $\hat{\mathbf{S}}^{(t)}$. As the cascade proceeds, we can be more certain that a precise semantic meaning on the face is being indexed. There are many more details that contribute to the relative speed and accuracy of the ERT method, such as how splits in each decision tree are computed, and how each regressor is trained, but they would take too long to cover in this report.

2.6.1 Implementation

We use two implementations for this project. Both utilize the Dlib image processing library. The first is Dlib's pretrained model. This model uses a cascade depth of 10, a tree depth of 4, 500 trees per cascade level, and a feature pool size of 400. It also has an oversampling amount of 20 for regularization. As you might expect, a deeper tree depth leads to a potentially more accurate regressor, but it is slower during prediction time. In addition, a deeper cascade depth can also lead to a much more accurate predictor, but it also creates a much larger output model.

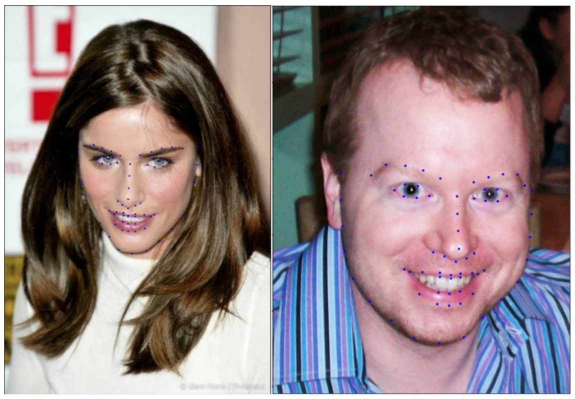
Figure 13. The pre-trained ERT model has excellent performance on a wide variety of faces, even with very different lighting conditions

The second implementation uses the same Dlib ERT model, but is trained from scratch on the 300-W large dataset. Importantly, we have modified it so that it only tries to fit the mouth. Our motivation for a lip-only model is for faster feature extraction, and more accurate lip matching, since the model no longer has to worry about matching the rest of the facial features. We increased the cascade depth to 15, and decreased the oversampling amount to 5 for less regularization. We also added a small amount of oversampling translation jitter, in order to apply some translation augmentation during training. These changes were made to compensate for a smaller training set than whatever the pre-trained model used.

2.6.2 Performance

ERT was designed with the goal of having extremely fast detection, so it is no surprise that the performance of both models is excellent. On average, extracting facial features from a typical 500x800 LFPW image took about .001-.002 seconds on the same 6-core i7 CPU used for the AAM and SDM tests. This is an order of magnitude faster than even SDM. Presumably, if we reduced the tree depth even further, the performance would be even better. Fitting time on the lip-only detector was more or less identical.

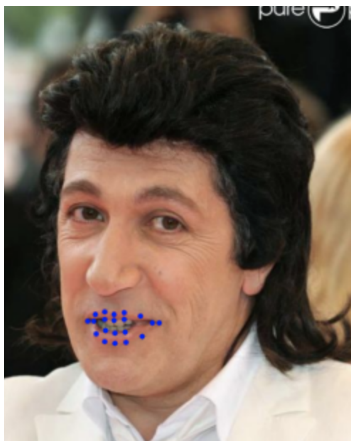
Figure 14. The mouth-only ERT model is also quite robust, despite the face detector providing less accurate initializations.

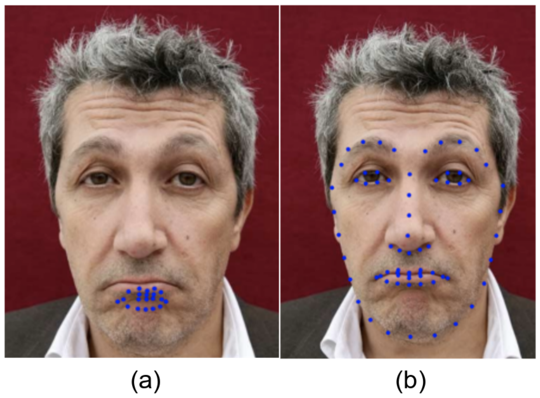
Figure 15. There are cases for which the full-face ERT model performs better than the mouth-only ERT model, potentially because of the more accurate initialization for the full-face model as well as there being fewer false positives for the whole face.

Since Dlib does not give us the minimum or maximum error values, we can only compare the mean error. Our mouth-only model has an error of 11.03. This is slightly better than our SDM model, but the more noticeable difference comes with the full-face model, which has an error of only 6.32. This performance gap can be explained by the fact that the added context of other facial features can prevent the model from fitting to misleading image features.

2.6.3 Examples

We can see from Figure 13 fitting result that the Dlib pre-trained ERT produces a nearly perfect fitting result. Just as for the prior models, the initialization is provided by Dlib's frontal face detector, although we no longer show the initial bounding box. This ERT detector is extremely robust, and even a wide variety of lighting conditions and faces, we could not produce a bad fitting result. From our testing, it is clear that the Dlib detector is both fast and accurate enough for use in the deep learning model.

The mouth-only detector generally has excellent performance, and in nearly every example image we tested was able to very closely match the subject's lips (see Figure 14). In theory, using the lip-only points in the deep learning model should be better, since the model won't get confused and try to learn from features like the nose, which do not affect speech.

For a few examples, like the one shown Figure 15a, we can see that the mouth-only detector gets confused. In this case, it is likely because the subject is making an exaggerated frown, which combined with his facial hair has created lots of shadows and misleading features beneath the mouth. As a result, the model has found a mouths-shaped wrinkle on the subject's chin.

Somewhat surprisingly, even though the mouth-only model was unable to correctly fit the subject's mouth, the whole-face model was able to perfectly outline the lips (see Figure 15). Although this could simply be a side-effect of the pre-trained model using more data, it is also likely that the whole-face model has a regularizing effect on the fitting result. When we try to only fit the mouth, our model may get confused by anything in the entire image that looks like a mouth, as in the above example. However, our whole-face model knows that in general, eyebrows must come above eyes, which must be above a nose, which is above a mouth. In other words, the whole-face model has context and knows the relative locations of each facial feature, making it more difficult to fit misleading image features.

2.7. Discussion

A comparison of the FFPD methods can be seen in Table 2.6.3. SDM and ERT are far and away the most accurate feature extraction methods. The accuracy for these two methods is orders of magnitude better than the older methods, though this is to be expected given how expensive they are to train and how much data they require to produce good results. Although the two have comparable accuracy, ERT still has better accuracy overall, and has the additional benefit of noticeably better performance. Although it is possible that this is influenced by a difference in their underlying implementations, Dlib's ERT implementation is the best candidate for our deep learning model for its fast extraction time and great accuracy. Unsurprisingly, we see a direct correlation between training time and model accuracy. Models like AAM were extremely quick to train and could produce good results even when trained on very little data, as long as a good initialization is provided. However, models like SDM and ERT are evidently more complex, as they require lots more data to provide reasonable results, and take much more time to train. However, once fully trained they are able to produce excellent results even when given a poor initialization. Furthermore, there seems to be a tradeoff between time spent training, and time spent during feature extraction, Again, ERT and SDM are much faster during feature extraction than the older models, despite taking much longer at training time.

The performance of both ACM and LACM suffered from requiring user input for initialization. It is challenging to select optimal parameters such that the lip outline is contoured. In addition, both models get confused by uneven lighting and deformities around the lips. Hence, both ACM and LACM without further pre-processing or other means of optimization are intractable for facial feature tracking and recognition.

3. Application for Automatic Lip Reading

In this section, we show a direct application of FFD for ARL of word utterances. We describe two DL frameworks: a feature-based approach and an end-to-end approach. Both models utilize FFD in order to pre-process video frames of the word utterances. Due to limited time and computational resources, we only utilize the ERT method outlined in Section 2.6 due to its high speed and accuracy. All models were implemented using Torch.

In order to train and evaluate the models, we utilize the LRW video database [39]. The database contains over 400,000 videos of 500 word utterances from over 1000 different speakers collected from BBC channels. Each video is fixed to be 29 frames in length with the word spoken in the middle of the segment. The database is already separated into training, validation and testing sets, following a split of approximately 1000/50/50 videos for each word. Due to the vast size of the database and our limited time and computational resources, we selected 20 words out of the 500 to use, which can be seen in Figure 18 and 19.

3.1. Feature-based approach

In order to show a simple and direct application of FFD for the ARL task, we design a simple deep learning model that utilizes a bi-directional RNN-LSTM [40]. An overview of the system pipeline is shown in Figure 16, and architecture hyperparameters are given Table 2. The 68-point coordinates from each video frame are extracted and normalized by fixing the central nose point (point 31 in Multi-PIE annotation scheme) as the axis origin. A categorical cross-entropy loss is used to train the model as a multi-classification problem for the 20 word utterances. Dropout rate of $p = 0.25$ and an L2-regularization coefficient of $\lambda = 10^{-6}$ are used during training with batch sizes of 128. We used the optimizer Adam [41] with a learning rate of 10^{-4}, and the default hyperparameters: a first-moment momentum coefficient of 0.9, a second-moment momentum coefficient of 0.999, and the numerical stability parameter $\epsilon = 10^{-8}$. Training was done for a total of 100 epochs, and the learning rate was decayed by 0.1 on the plateau of the validation loss.

3.2. End-to-end approach

State-of-the-art solutions for the ARL task utilize an end-to-end DL architecture, where the video is directly placed into the pipeline. One of the first successful end-to-end approaches to tackle the sentence-level task and surpass human-expert performance was LipNet [36]. We implement LipNet with a few minor modifications: batch normalization layers [42] are added after each convolutional layers, and the CTC loss is replaced with a categorical cross-

Method	# of Landmarks	Error (Mean Euclidean Distance)	Computation Time (s)
ASM (CLM)	68	38.4	334.0
AAM	68	427	113.6
SDM	68	19.9	11.04
ERT (mouth only)	19	11.03	29.5
ERT (whole face)	68	6.32	6.32

Table 1. Comparison of FFPD methods using the LFPW database test set.

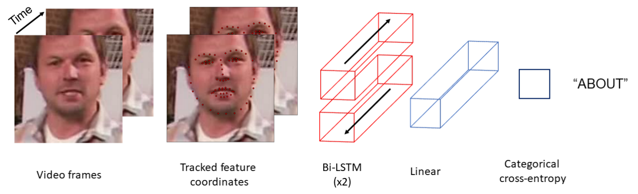

Figure 16. Feature-only network architecture. 68-point coordinates from each video frame are extracted using FFD, normalized by fixing the central nose point (point 31 in Multi-PIE annotation scheme) as the axis origin, and flattened into a single dimension. The sequence of coordinates are then fed into a 2-layer Bi-LSTM. The weights from the final Bi-LSTM layer are then concatenated and processed by a linear layer and a softmax. This model is trained with categorical cross-entropy loss.

entropy loss as single words are predicted as opposed to sentence sequences. An overview of the pipeline is shown in Figure 17. Detected mouth feature points from the video are used to extract a mouth-centred crop of size 100×50 pixels per frame, which are then fed into the model. Dropout rate of $p = 0.5$ and an L2-regularization coefficient of $\lambda = 10^{-6}$ are used during training with batch sizes of 64. The optimizer Adam is used with a learning rate of 10^{-4}, and the default hyperparameters: a first-moment momentum coefficient of 0.9, a second-moment momentum coefficient of 0.999, and the numerical stability parameter $\epsilon = 10^{-8}$. Training was done for a total of 100 epochs, and the learning rate was decayed by 0.1 on the plateau of the validation loss.

3.3. Results

The results for the models are shown as a confusion matrices in Figure 18 and 19. The feature-based model manages to achieve an overall testing accuracy 72.1% is

Layer	Hidden	Input	Dimension
Bi-LSTM	512	$29 \times (68 \times 2)$	$T \times (F \times 2)$
Bi-LSTM	512	29×512	$T \times H$
Linear	20	(512×2)	$(H \times 2)$
Softmax		20	K

Table 2. Facial feature network architecture hyperparameters. T is the number of frames, F is the number of facial feature points, H is the hidden dimension size, and K is the number of word labels.

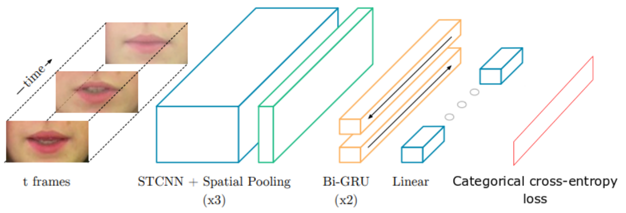

Figure 17. LipNet architecture. A sequence of T mouth-centered crop frames is used as input to a 3-layer spatio-temporal convolutional neural network (STCNN), where each layer is followed by batch normalization, ReLU activation and spatial max-pooling layers. The features extracted from the STCNN are then processed by 2 Bi-GRUs. The final time-step of the GRU output is then processed by a linear layer and a softmax. This end-to-end model is trained with categorical cross-entropy loss.

achieved, while the end-to-end model manages to achieve an overall testing accuracy of 90.3%. The superior performance of LipNet is attributed to the fact that the network can learn both visual and temporal features together from the video frames directly as opposed to only learning temporal features in the feature-based approach.

4. Conclusion

In this report, we have surveyed 6 FFD techniques, of which the 4 FFPD methods have be quantitatively compared using the LFPW database test set. We find that the discrimative CRM methods SDM and ERT perform orders of magnitude better than the generative deformable model methods ASM and AAM, both in terms of accuracy and computation time. Nevertheless, ASM and AAM are able to obtain adequate results with fast training time and good initialization. However, ASM remained the slowest during feature extraction time. The snake based FFD methods discussed were shown to struggle with uneven lighting, lip deformities, long computation times, and user input for initialization, making them unsuitable for the task of mouth or lip recognition and segmentation.

Finally, we have shown a successful application of FFD for the task of ARL using 2 separate DL approaches. We show that a simple facial feature-only network can learn to classify 20 words adequately, and that an end-to-end network can achieve over 90% accuracy. In the future, we propose that a multi-modal fusion network could be used to join the facial feature-only and end-to-end to networks. A possible benefit of such an architecture is enabling the use of a deeper STCNN in the end-to-end network, which previously suffers from overfitting.

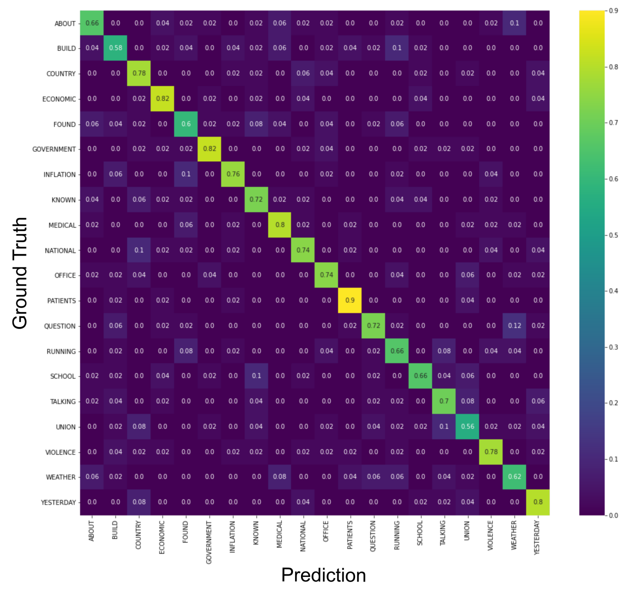

Figure 18. Confusion matrix results for facial feature-only network.

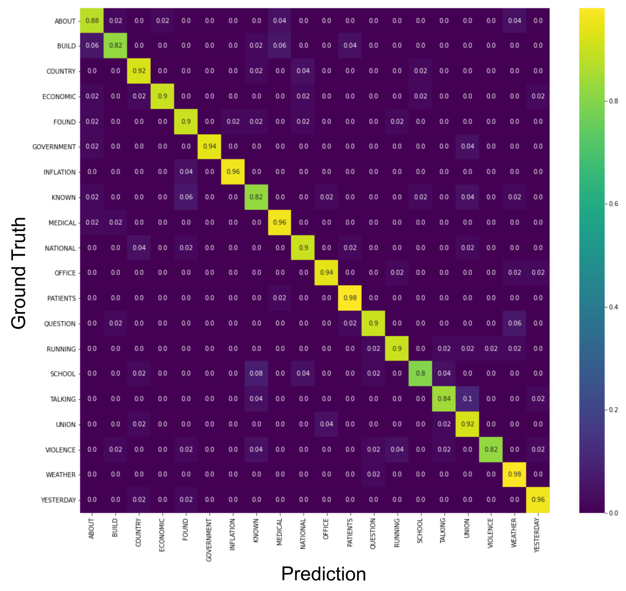

Figure 19. Confusion matrix results for LipNet.

References

[1] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 4, pp. 607–626, 2008.

[2] Y.-L. Tian, T. Kanade, and J. F. Cohn, "Facial expression analysis," in *Handbook of face recognition*, pp. 247–275, Springer, 2005.

[3] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Comput. Surv.*, vol. 35, p. 399–458, Dec. 2003.

[4] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," *ACM Trans. Graph.*, vol. 30, July 2011.

[5] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, (USA), p. 187–194, ACM Press/Addison-Wesley Publishing Co., 1999.

[6] I. Matthews, T. F. Cootes, J. A. Bangham, S. Cox, and R. Harvey, "Extraction of visual features for lipreading," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 198–213, 2002.

[7] A. Fernandez-Lopez and F. M. Sukno, "Survey on automatic lip-reading in the era of deep learning," *Image and Vision Computing*, vol. 78, pp. 53–72, 2018.

[8] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pp. 2144–2151, IEEE, 2011.

[9] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, "Localizing parts of faces using a consensus of exemplars," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2930–2940, 2013.

[10] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 397–403, 2013.

[11] C. Tomasi and T. Kanade, "Detection and tracking of point features," tech. rep., International Journal of Computer Vision, 1991.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, p. 91–110, Nov. 2004.

[13] Li Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[14] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1465–1479, 2006.

[15] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, p. 448—461, March 2010.

[16] P. I. Wilson and J. Fernandez, "Facial feature detection using haar classifiers," *J. Comput. Sci. Coll.*, vol. 21, p. 127–133, Apr. 2006.

[17] H. Wu, T. Yokoyama, D. Pramadihanto, and M. Yachida, "Face and facial feature extraction from color image," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 345–350, 1996.

[18] A. Nikolaidis and I. Pitas, "Facial feature extraction and pose determination," *Pattern Recognition*, vol. 33, no. 11, pp. 1783 – 1791, 2000.

[19] U. Saeed and J.-L. Dugelay, "Combining edge detection and region segmentation for lip contour extraction," in *Articulated Motion and Deformable Objects* (F. J. Perales and R. B. Fisher, eds.), (Berlin, Heidelberg), pp. 11–20, Springer Berlin Heidelberg, 2010.

[20] T. F. Cootes and C. J. Taylor, "Active shape models—'smart snakes'," in *BMVC92*, pp. 266–275, Springer, 1992.

[21] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[22] D. Cristinacce and T. F. Cootes, "Feature detection and tracking with constrained local models.," in *Bmvc*, vol. 1, p. 3, Citeseer, 2006.

[23] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 532–539, 2013.

[24] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1867–1874, 2014.

[25] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, 2014.

[26] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3476–3483, 2013.

[27] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 94–108, Springer International Publishing, 2014.

[28] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.

[29] K. Sobottka and I. Pitas, "Segmentation and tracking of faces in color images," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 236–241, 1996.

[30] X. Liu and Y. Cheung, "A robust lip tracking algorithm using localized color active contours and deformable models," pp. 1197–1200, 2011.

[31] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "A semi-automatic methodology for facial landmark annotation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 896–903, 2013.

[32] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: Database and results," *Image and vision computing*, vol. 47, pp. 3–18, 2016.

[33] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-pie," *Image and Vision Computing*, vol. 28, no. 5, pp. 807 – 813, 2010. Best of Automatic Face and Gesture Recognition 2008.

[34] R. Easton and M. Basala, "Perceptual dominance during lipreading," *Perception Psychophysics*, vol. 32, pp. 562–570, 11 1982.

[35] C. G. Fisher, "Confusions among visually perceived consonants," *Journal of Speech and Hearing Research*, vol. 11, no. 4, pp. 796–804, 1968.

[36] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, "Lipnet: End-to-end sentence-level lipreading," 2016.

[37] J. Luettin, N. A. Thacker, and S. W. Beet, "Visual speech recognition using active shape models and hidden markov models," vol. 2, pp. 817–820 vol. 2, 1996.

[38] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[39] J. S. Chung and A. Zisserman, "Lip reading in the wild," in *Computer Vision – ACCV 2016* (S.-H. Lai, V. Lepetit, K. Nishino, and Y. Sato, eds.), (Cham), pp. 87–103, Springer International Publishing, 2017.

[40] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, pp. 2047–2052, IEEE, 2005.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[42] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.