Human Skin Segmentation in Videos with Fully Convolutional Neural Networks

Gabriel Birman

Department of Computer Science, Princeton University

Abstract

Convolutional Neural Networks (CNN) have become the state-of-the-art in many computer vision applications, including the task of skin detection in humans. With reference to the task of skin segmentation, this paper not only improves the performance of the current state-of-the-art CNN in the spatial dimension, but also demonstrates increased performance when considering the temporal dimension – relevant for improved skin detection in video formats. The proposed multi-frame CNN refines the output of an arbitrary pre-trained skin detector yielding a small performance boost under specific conditions, and can readily be applied to pre-recorded and real-time skin segmentation.

1. Introduction

Skin detection is the problem of labelling pixels in images as either belong to human skin or not belonging to human skin. This often serves as an intermediate processing step for image enhancement, face and human detection, gesture analysis, pornographic contents filtering, surveillance systems, etc. [1]

In the past couple decades, many solutions have emerged to address the first problem, namely skin detection in the spatial dimension, with CNNs emerging in recent years as the state-of-the-art. To note, two recent papers have presented a CNN-based approach to skin detection. The first paper explored the performance of VGG-based [2] and NiN-based systems [3], ultimately concluding that a whole-image NiN CNN provides the best results [1]. The second paper uses a ResNet-based architecture [4] with similar results [5].

By the nature of the aforementioned applications of skin detection, it is clear that much of the relevant data exists in video form. Given the challenges of developing a high-performance skin detector in images such as complex background, diverse light conditions, and cluttered environment [5], it is possible that taking advantage of the temporal dimension would improve performance without proving too costly in terms of inference speed.

Although there have been some attempts to improve video skin segmentation with color-based methods [6, 7], and there has been work showing performance improvement with a temporal CNN [8], there have been no work exclusively dedicated to the topic of combining the skin segmentation problem with a temporal CNN (to the best of the author's knowledge).

In this paper, the author presents an improved CNN for skin detection as an extension to the network of [1, 3], where the effects of a Euclidean loss and Cross-Entropy loss on training the network, and the effects of the evaluation methodology are considered. Then, the author incorporates this CNN into a full-fledged Temporal Refinement Network that uses a set of images temporally centered around an input image to detect skin in that image. This network demonstrates improved results in some facets, yet needs further improvement in other facets. Ultimately,

2. Proposed Method

This Temporal Refinement Network is broken down into two parts. Firstly, a suitable base network is learned for inference on color images. Use of grayscale images was also considered at one point, but network performance was worse on the validation set, which mirrors the results of [1]. Then, the base network is used to train the refinement module for simplicity. Together, these comprise the Temporal Refinement Network.

2.1. Base Network

The base network was originally conceived to be either the image-based NiN network [1, 3] or the ResNet network [5]. Ultimately, the image-based NiN network was chosen as the base network. The rationale for this was that although the ResNet network provides slightly improved accuracy on the test set, the F1 score of the NiN network was much larger [5], leading to the conclusion that the image-based NiN network provided the best overall skin detector among the evaluated networks. The image-based NiN network structure, loss function, and input pre-processing is then slightly modified to achieve improved results. The modified network's performance (with and without the inclusion of the new loss

function) is then compared to the original NiN network performance.

2.1.1 Network Structure

The image-based NiN network can be divided into four parts. The first part consists of 3 convolution layers of 7x7, 1x1 and 3x3 kernels. The second part is a set of modified inception modules which are described in section 3.1 of [3] where all modules are identical. The number of modified inception modules is set to 8. Then, one convolution layer is used to integrate all the feature maps in a single channel which is a skin map produced by the proposed method. In the paper, a Euclidean loss function is included for the training phase, but it is modified as described below. The reader is asked to refer to the NiN paper [3] for further detail on the network structure and relevant figures.

Unfortunately, no code was provided for [3, 1], so a few assumptions had to be made and/or deduced when reconstructing their network. For instance, it was unclear whether biases were included in the convolutions. Nevertheless, the number of network parameters of the original NiN network (236K) [1] suggests that each convolutional layer included a bias term, as removing these produced a smaller number of network parameters. Moreover, this would suggest that batchnorm was not originally included as this would raise the number of network parameters.

For the purposes of the base network, only the final convolutional layer included a bias term. Moreover, batch-normalization [9] was included in every convolutional layer but the last.

2.1.2 Loss Function

The original loss function is a Euclidean loss function commonly used in regression tasks. Namely, given training images and groundtruth images $\{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^N$, the loss function is:

$$\mathbf{L}(\Theta) = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{Y}_i - F(\mathbf{X}_i;\Theta)\|^2,$$

where N is the number of images and Θ is the parameters of the network [3]. Given that the output of the network is a probability map, and the target images $\mathbf{Y}_i$ are composed of binary values 1 and 0 representing skin and non-skin pixels, respectively, it was proposed that a binary cross-entropy loss function with sigmoid activation be used instead:

$$\mathbf{L}(\Theta) = -\frac{1}{N}\sum_{i=1}^{N}\mathbf{Y}_i \log\sigma(F(\mathbf{X}_i;\Theta)) + (1-\mathbf{Y}_i)\log\left(1-\sigma(F(\mathbf{X}_i;\Theta))\right),$$

where σ represents the sigmoid activation function used to output probability values in the range $(0, 1)$.

2.1.3 Inputs and Outputs

A few changes were made to input pre-processing. Instead of subtracting the input images by the average intensity of the training dataset [3], inputs were normalized channel-wise to have mean 0 and standard deviation 1 across the three R, G, B channels according to the training dataset mean and standard deviation.

In order to address the problem of various sizes and aspect ratios, this paper performs the steps described in [3]. Namely, during training, the input image is resized using bilinear interpolation such that smaller side (horizontal/vertical) is reduced to 50, and then the image is stridden in the direction of the larger side to obtain a batch of 50x50 images. The groundtruth images are preprocessed in the same way; however, when training the cross entropy loss function, groundtruth images are instead downsampled with nearest neighbor interpolation to preserve the binary range.

There is some discrepancy to how outputs are resized during inference time. [3] claims that the downsized probability maps are simply interpolated to original size, whereas [1] claims that the stridden probability maps are merged and interpolated to the original size. This discrepancy is especially salient given that both papers report the same results. It is possible, then, that evaluation was not performed on the interpolated images, but rather the downsized images.

For the purposes of this paper, the evaluation was performed in the following way: the input images were downsized (bilinear interpolation) such that the smaller side was length 50; then, the predicted probability maps were upsampled (bilinear interpolation) to the original size and compared with the original groundtruth data. This ensures that the evaluation is not subject to the interpolation method of the groundtruth data, and has the added benefit of smoothing the predicted map.

2.1.4 Training and Testing

The hyperparameters provided in the original paper, such as number of inception modules, weight decay etc. were not re-tested as it was assumed that the author's conclusion that these represented the best combination was scientifically

sound. The exception to this rule was the learning rate and training batch size hyper-parameters, which were chosen by cross validation.

Instead of initializing weights to zero as in [3], weights were generated from a truncated normal distribution with lower bound -2 and upper bound 2, then scaled down by a factor of 100. The trainable bias parameter of the final convolutional layer was initialized to 0 when using the cross-entropy loss, and 0.5 when using the Euclidean loss, such that the probability of classifying a skin pixel was roughly equal to the probability of classifying a non-skin pixel at initialization.

Note that during testing and inference time, the sigmoid activation is applied to the network output to obtain the probability map, only when considering the network trained with the cross entropy loss function with logits.

2.2. Temporal Refinement Network

The Temporal Refinement Network, as described below, is trained with pre-processed video data and evaluated across several different hyperparameters. Its performance is then tested against the Base Network.

2.2.1 Network Structure

The Temporal Refinement Network consists of the base network and the refinement module. To avoid complicating the testing procedure, the refinement module has the same network structure as the base network but a different number of input channels (namely, T input channels, as described in the "Inputs and Outputs" section below).

The performance of the refinement module is also considered with and without a skip architecture. Specifically, with F_0 representing the pre-trained base network function, and F_1 representing an identical refinement module (distinguished simply by number of input channels), the temporal refinement network architecture without the skip is: $F_1(F_0(\mathbf{X}_{i;t};\Theta_0);\Theta_1)$; whereas, with the skip it is: $F_0(\mathbf{X}_{i;t};\Theta_0) + F_1(F_0(\mathbf{X}_{i;t};\Theta_0);\Theta_1)$. The consideration of the skip architecture is guided by the principles of ResNet, which shows that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping, i.e. it is easier to improve on the identity by learning the residual separately [4]. Figure 1 provides a visualization of the entire network pipeline with the skip architecture.

2.2.2 Loss Function

The loss function that produced the best results on the base network was used for the refinement module. This happened to be the cross-entropy loss function with sigmoid activation.

2.2.3 Inputs and Outputs

The Temporal Refinement Network takes as input a set of temporal input images derived from video data. Assuming that our video data contains N frames $\{\mathbf{X}_{i;t}\}_{i=1}^{N}$ captured at some time t, where t can vary for each frame, then the derived temporal input images would be centered around $\{\mathbf{X}_{i;t}\}_{i=1}^{N}$. Each temporal input image is run through the Base Network to produce a probability map. These probability maps are then concatenated in temporal order and run through the refinement module to produce a single probability map for the input image $\{\mathbf{X}_{i;t}\}_{i=1}^{N}$.

Note that after the relevant video frames are consolidated as input images, the input images and groundtruth images are pre-processed the same as before.

Figure 2: *The input image (in red) alongside its temporal image set for $T = 7$. Images are chronologically sorted in left to right, top to bottom order.*

2.2.4 Training and Testing

For the purposes of training and testing our network, the temporal input set is delineated by two parameters, T and δ, where T is an odd number representing the number of temporal input images, and δ is the uniform time interval between them.

Thus, given input images and groundtruth images for video data $\{\mathbf{X}_{i;t}, \mathbf{Y}_{i;t}\}_{i=1}^{N}$, we produce a set of temporal input images $\{\mathbf{X}_{i;t-\delta\frac{T-1}{2}}, \ldots, \mathbf{X}_{i;t-\delta}, \mathbf{X}_{i;t}, \mathbf{X}_{i;t+\delta}, \ldots, \mathbf{X}_{i;t+\delta\frac{T-1}{2}}\}_{i=1}^{N}$ centered around the input-groundtruth pair $\{\mathbf{X}_{i;t}, \mathbf{Y}_{i;t}\}_{i=1}^{N}$ for a given T and δ. Figure 2 shows the temporal set for $T = 7$ and some δ.

Each input image produces R temporal image subsets to be added to the total training/testing set. Each of these uses

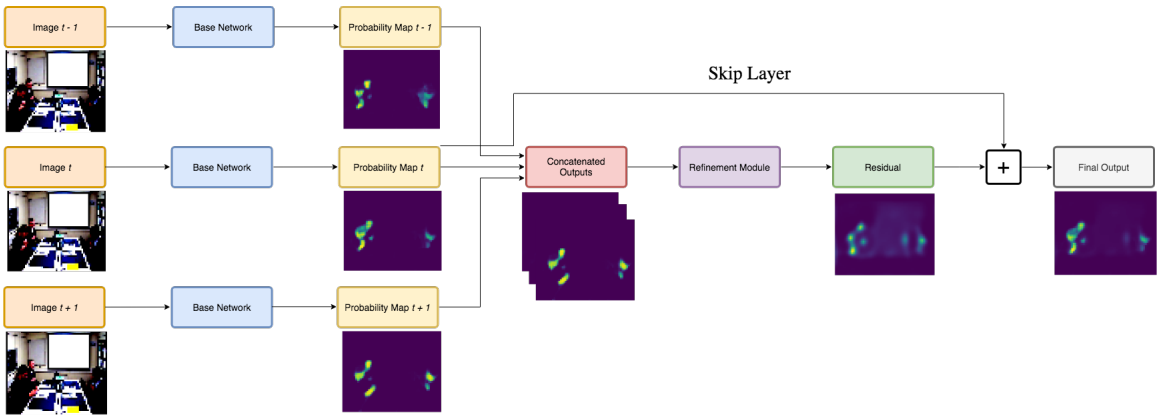

Figure 1: *Showing the pipeline for the Temporal Refinement Network with the skip architecture on a sample image set ($T = 3$).*

random delta uniformly chosen in $\delta = (\delta_{lo}, \delta_{hi})$. Note that when $T = 1$, only one input image is included in the training set as it would be spurious to produce many identical images.

The network hyperparameters and parameters are initialized the same as before, except that the bias for the residual network is set to zero. Notably, the base network parameters are frozen during the training of the Temporal Refinement Network so as to solely train the refinement module itself to take advantage of temporal information.

3. Experiments

3.1. Experimental Setup

3.1.1 Base Network

PyTorch [10] was used to train the networks. The networks were trained using four 2080Ti GPUs with 11GB of memory each. The Adam optimizer [11] was used with the following parameters: learning rate = 0.001, weight decay = 0.0002, β_1 = 0.9, β_2 = 0.999, and $\epsilon = 1 \times 10^{-8}$. Cross-validation on the FSD dataset [12] was performed on the learning rate and training batch size parameters, where it was determined that 0.001 and 32 gave the best performance, respectively. Testing was done with a batch size of 1 due to the varying aspect ratios of images, as these were not stridden to size 50x50.

In addition, a learning rate scheduler was used that decreased the learning rate upon stagnating validation loss (the default settings were used for PyTorch's ReduceLROnPlateau learning rate scheduler). The training exited early when validation loss had not improved for 25 epochs, and the parameters that provided the best validation loss were used.

When training with the modified NiN network (Euclidean), it took approx. 3 hours to train 273 epochs. When training with the modified NiN network (Cross-Entropy), it took approx. 2 hours to train 196 epochs.

The base network was trained on the FSD dataset [12] and evaluated on the Pratheepan (facial) dataset [13], as in [1, 5].

3.1.2 Temporal Refinement Network

The Temporal Refinement Network was trained with the same optimizer/learning rate scheduler parameters as before, but the weight decay was set to 0.01, a value determined by cross validation on the baseline $T = 1$.

The network relied on the VDM dataset [14] for labelled images, in particular using the labelled LIRIS dataset [15] for training and the labelled AMI dataset [16] for testing, as these were two readily available datasets. The details are described in the "Datasets" subsection.

The Temporal Refinement Network was trained/tested with $T = 1, 3, 5, 7$, $R = 10$, with each temporal image set being defined by a specific $\delta \in (0.25, 5)$ (seconds).

These parameters are arbitrarily chosen, but with good measure. It is hypothesized that the refinement module would work by capturing and learning from motion information in the data. The upper and lower bounds provide reasonable time frames for capturing a certain degree of motion. Too little motion and the frames would be quasi-identitical, making the temporal input no better than the spatial input. Too much motion, and the frames would be hardly recognizable, making it hard to take advantage of similarity. The number of training sets, $R = 10$, reduces variance across multiple samples of delta and hopefully ensures that the network has sufficient information to adequately train on a variety of time intervals.

As the video frames of both the AMI and LIRIS dataset were captured at discrete time intervals at a rate of 25 frames per second [16, 17], this corresponded to a frame offset between 8 frames and 125 frames. When a sampled δ produced

an incoherent result, i.e. the selected time interval was out of bounds, the δ value would be resampled. The pre-processor would continue on to the next input image after 100 such incoherent results.

Figure 3: *Sample images from the FSD, Pratheepan, LIRIS, and AMI datasets (left to right)*

Figure 4: *A comparison between the input image in the VDM-LIRIS training dataset (left) and the matched image in the original LIRIS dataset (right).*

3.2. Datasets

3.2.1 Base Network

The FSD dataset is composed of 4000 colour images across various lighting conditions, occlusion conditions, and human skintones, though it primarily consisted of single-person images. The dataset was calculated to have channel-wise mean 102.9, 111.5, 126.3 and channel-wise standard deviation 72.2, 70.6, 74.8 (in B, G, R order), so these numbers were used to normalize the data. The FSD dataset was trained with 20 percent of the data used for validation, where the validation-training indices were selected at random. Note that no data from the FSD dataset was used for final evaluation unlike in [1, 3].

The Pratheepan dataset contains 78 high resolution images of various skintones and lighting conditions, and was split into two separate datasets. The first dataset contained 32 facial images, i.e. faces of a single person in each photo. The second dataset contained 46 family photos, i.e. photos containing humans in a variety of positions, which made it significantly harder for skin detection.

3.2.2 Temporal Refinement Network

The VDM dataset provides groundtruth images for a subset of video frames from various human activity recognition datasets, including the AMI dataset, which contains 100 hours of recorded meetings, and the LIRIS dataset, which contains videos of people performing various activities common during daily life. As the VDM authors did not include their indexing methods, the input-groundtruth labels were matched to the corresponding video frame in the respective dataset by considering the video frame that minimized the normed MSE loss between the two; it was not possible to simply check for image matrix equality as artifacts had been introduced images during the construction of VDM dataset.

For the AMI dataset, it was observed that the frames were based on non-scenario videos taken in the IDIAP Room. As opposed to minimizing the loss across all the possible videos, a one-to-one matching between input image and the video it was extracted from was determined by matching the appearances of the people in the room and the content of the slideshow on the projector. As there were too many possible source videos for the LIRIS dataset, the loss minimization was performed across all videos (excluding grayscale videos). A qualitative evaluation of the matched images suggests that the loss minimization produced a perfect matching (or close enough to perfect such that the two images are indistinguishable). Figure 4 demonstrates this successful matching.

The LIRIS dataset was used for training as it contained different lighting conditions and video locations, whereas the AMI dataset was shot in the same room for each video, with little variation in the video information. The RGB portion of the LIRIS dataset was split along pre-established training and testing splits, consisting of 30 images and 25 images respectively. 30 percent of the images in the test set were then used for validation. All of the relevant portions of the AMI corpus were used for evaluation.

Table 1: *Base Network Evaluation Results on Pratheepan (Facial) Dataset at Peak F-measure*

Methods	Accuracy	Precision	Recall	F-measure
Image-NiN [1]	0.9484	0.9003	0.8912	0.8957
ResNet [5]	0.9499	0.8480	0.8981	0.8678
Ours (CE)	0.9312	0.8409	0.8875	0.8613
Ours (Eucl.)	**0.9553**	**0.9022**	**0.9108**	**0.9045**

Table 2: *Comparison of Base Network Evaluation Methods (Upsampling Prediction vs. Downsampling Groundtruth) on Pratheepan (Facial) Dataset at Peak F-measure*

Methods	Accuracy	Precision	Recall	F-measure
Up (CE)	0.9312	0.8409	0.8875	0.8613
Up (Eucl.)	**0.9553**	**0.9022**	0.9108	**0.9045**
Down (CE)	0.9483	0.8846	**0.9120**	0.8957
Down (Eucl.)	0.9449	0.8747	0.8934	0.8822

Figure 5: *Showing the results of the Base Network on a sample image. The input image (top-left), groundtruth image (top-right), prediction (bottom-left), and binarized image (bottom-right) are all showcased.*

3.3. Experimental Results

3.3.1 Base Network

Evaluation was done by comparing the Precision, Accuracy, Recall, and F1-score at peak F-measure, as in [1, 5]. Though it is unclear exactly how the peak F-measure was obtained in the above, in this paper the peak F-measure was obtained by considering the maximum F-1 score across a binarization threshold grid ranging from 0 to 1 with step size 0.01. Figure **??** shows the base network output binarized with the threshold that would maximize F-measure.

The base network was evaluated on the facial subset of the Pratheepan dataset as in [3]. Notably, the ResNet skin-detector authors claim that their network was tested on the entire Pratheepan dataset [5]. However, as this paper's trained networks performed substantially worse on the entire dataset, yet demonstrated similar results to [3] on the facial dataset, the author conjectures that the [5] network was in fact evaluated on the facial dataset, though no attempt was made to empirically verify this hypothesis.

Table 1 shows that the modified NiN network underperformed the original NiN network with the cross entropy loss function, but demonstrated improved performance with the addition of weight initialization, data pre-processing, and batch normalization, though it is not clear how much each of these factors influenced the outcome.

The same evaluation was also performed with a different methodology: instead of upsampling the prediction with bilinear interpolation, the groundtruth was instead downsampled using nearest neighbor interpolation. Table 2 compares the evaluation methods on the Pratheepan Facial dataset, and shows an interesting reversal. While the upsampling evaluation performs better with Euclidean loss, downsampling evaluation performs better with cross-entropy loss. Moreover, upsampling improves both precision and recall of Euclidean loss network, while decreasing both precision and recall of Cross-Entropy loss network (at peak F-measure). The author hypothesizes that this phenomenon is the result of the validation method. With Euclidean loss, the groundtruth was downsampled with bilinear interpolation, whereas with Cross-Entropy loss, the groundtruth was downsampled with nearest neighbor interpolation. As a result, the Euclidean loss network was better tuned to bilinear upsampling, and the groundtruth was better tuned to nearest neighbor downsampling. This phenomenon shows that a thorough understanding and uniform usage of evaluation methods is necessary when comparing the performance of skin detection networks. Ultimately, it cannot be completely certain which network performs best as the details of the evaluation methods of [1] and [5] were not well-known.

Another limitation of the base network evaluation was that no data was held out for evaluation on the FSD dataset. The authors of [3] claim that half the images were used for testing. In this paper, 20 percent of the images were used for validation, but none for testing. As a result, the performance improvements of the modified NiN network could simply be a result of more comprehensive training data.

3.3.2 Temporal Refinement Network

Due to the high performance of the modified Euclidean loss network, it was chosen to serve as the base network for the Temporal Refinement Network. One of the limitations of the Euclidean network is that there is no theoretical bound on the outputs. In fact, using the skip architecture, the refinement module learned large negative residuals which often resulted in negative "probabilities" in the output. The minimum threshold was lowered to -0.5 to account for this.

The results (prediction upsampling) are summarized in Table 3. Disappointingly, in all cases except the LIRIS $T = 7$ case, the base network outperforms the Temporal Refinement Network in terms of F1-score, a joint measure of recall and precision. And yet, the low F1-score for the baseline $T = 1$ is consistently improved upon with additional T, as a general trend, though there seems to be a dip around $T = 5$, which merits further exploration. It appears that the FSD dataset used for training the base network does not transfer well to these datasets, which by and large are low-resolution images that contain much less skin in each frame than the FSD dataset; combined with poor lighting conditions, and

often have people turned away from the camera (FSD almost always had people facing the camera), these datasets proved especially difficult for skin detection. This is corroborated by the simultaneous high accuracy and low F1-score of the results: the model has learned to set nearly every pixel to non-skin. The lack of true positives in the LIRIS training data, and the lack of training data itself, thus it difficult to learn a proper refinement module.

During training, the training loss and validation loss hovered around 0.01 MSE, an average pixel value discrepancy of 0.1 between prediction and groundtruth. Though this is a seemingly good result for the validation set, it can easily be achieved by more-or-less indiscriminately setting pixels to zero (non-skin), as the number of non-skin pixels in an image tended to greatly outweight the number of skin pixels. This explains the extremely poor recall of the baseline $T = 1$ on the test set in Table 3. To counterbalance this, the loss function could be modified to penalize false negatives more than false positives.

Importantly, the Temporal Refinement Network is primarily constrained by the base network, for it takes as input the probability map outputs that have been fed into the base network. The base network's performance on LIRIS and AMI, though seemingly better than the Temporal Refinement Network, is still quite poor. This, in turn, makes it difficult to evaluate the refinement network on its own, as the Temporal Refinement Network then has difficulty improving on inputs when the inputs themselves are too distorted to make sense of. For instance, Figure 6 shows that no improvement can be made with poor inputs. On the other hand, Figure 7 is an example of an improvement that can be made with the Temporal Refinement Network. With increased T, the refinement network not only learns to remove the table from the from the skin segmentation, but also better encodes the torso details of the left-most person, who was only represented by a head in the base network output.

The refinement module was also assessed with and without a skip architecture. The skip architecture performed slightly worse than without on both LIRIS and AMI, although this is possibly just the result of an initially poor identity (i.e. the base network output).

The author hypothesizes that the results would be much improved if the base network were sufficiently trained to perform well on the LIRIS dataset, as more coherent inputs would lead to better results. This hypothesis was put to the test by using transfer learning to train the Base Network on the LIRIS dataset. The peak F-measure improved from 0.2719 to 0.4520 on the LIRIS test set. In turn, the Temporal Base Network ($T = 3$, skip) had an improvement from 0.2713 to 0.4630. Though this did not outright confirm the hypothesis, it indicated that it might have some merit.

Even better would be to train a network to use multiple color images as input, but this could be prohibitively expensive to train, as the number of parameters would scale by the number of images; for $T = 7$, this network would have around 1.6M trainable parameters compared to the 250K of the current refinement module. When looking at inference time of a forward pass on a single 2.7 GHz Intel Core i5 CPU, it took the base network around 0.8 seconds. The refinement module took between 0.05 to 0.08 seconds for inference for all T values, so the base network is clearly the limiting factor, meaning that one could reasonably expect to train the refinement module on much larger T without greatly impacting the inference time, assuming that preprocessing steps are not too time-consuming. Whereas using a refinement network that scales linearly with the base network would clearly cause significant delays in inference time. When evaluating on real-time video data, this is an important consideration.

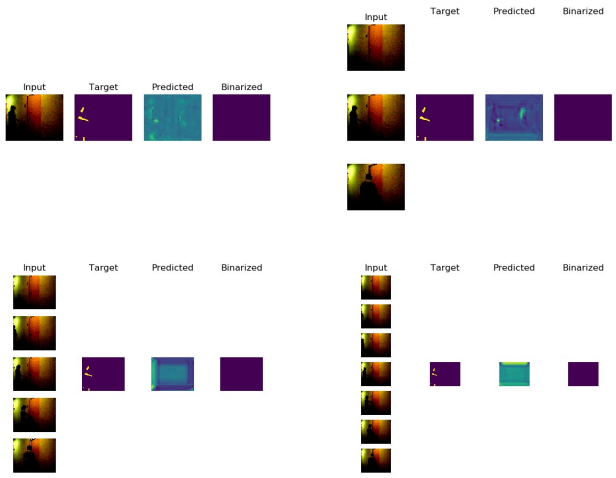

Figure 6: *Evaluation results of the Temporal Refinement Network (with Skip Architecture) on a sample image displaying poor performance. Base Network (top-left) is compared to $T = 3$ (top-right), $T = 5$, (bottom-left), $T = 7$ (bottom-right).*

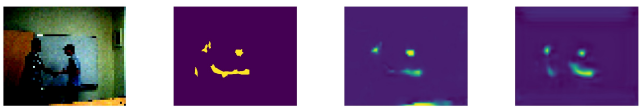

Figure 7: *Comparison of the Base Network output and the Temporal Refinement Network ($T = 7$, skip) output showing marked improvement. Images shown are input, groundtruth, base network, refinement network from left to right.*

Table 3: *Temporal Refinement Network Results for LIRIS (top) and AMI (bottom) Datasets at Peak F-measure*

Methods	Accuracy	Precision	Recall	F-measure	Threshold
Base Network	0.9741	0.3249	**0.3212**	0.2719	0.17
Skip (T=1)	0.9705	0.2312	0.3104	0.2180	-0.39
Skip (T=3)	0.9797	0.3315	0.2973	0.2713	-0.33
Skip (T=5)	**0.9825**	**0.4362**	0.2526	0.2580	-0.30
Skip (T=7)	0.9795	0.3622	0.3119	0.2816	-0.37
No Skip (T=1)	0.9723	0.2523	0.2926	0.2283	0.14
No Skip (T=3)	0.9801	0.2906	0.2809	0.2484	0.24
No Skip (T=5)	0.9803	0.3214	0.2723	0.2515	0.23
No Skip (T=7)	0.9810	0.4130	0.3042	**0.2871**	0.16
Base Network	0.9712	**0.5113**	0.5461	**0.5030**	0.26
Skip (T=1)	0.9695	0.4347	0.4263	0.4133	-0.27
Skip (T=3)	0.9714	0.4692	0.4781	0.4563	-0.29
Skip (T=5)	0.9717	0.4772	0.4744	0.4558	-0.30
Skip (T=7)	0.9728	0.4821	0.5007	0.4729	-0.33
No Skip (T=1)	0.9686	0.4398	0.4300	0.4170	0.27
No Skip (T=3)	0.9727	0.4915	0.5180	0.4867	0.28
No Skip (T=5)	**0.9730**	0.5010	0.4390	0.4492	0.30
No Skip (T=7)	0.9723	0.4865	**0.5463**	0.4975	0.22

4. Conclusion

By modifiying the Image-based NiN network of [1] and some steps involved in training it on the FSD dataset, this paper has achieved state-of-the-art results on the Pratheepan dataset. Moreover, it has been shown that the evaluation method can lead to large variation in outcomes; the Euclidean loss network performed better when upsampling the prediction (achieving the state-of-the-art), whereas the Cross-Entropy loss network performed when downsampling the groundtruth. A clearer standard for evaluation skin detection is needed to better compare network performance.

As for the Temporal Refinement Network, its performance was overall similar to the Base Network, which was unsurprising given the construction of the architecture to include the base network output as input. The complexity of the video datasets proved a limiting factor when assess the Temporal Refinement Network, as the refinement module's performance was constrained by the Base Network's poor performance on these data.

Nevertheless, the Temporal Refinement Network ($T = 7$) outperformed the Base Network on the LIRIS dataset, and slightly underperformed the Base Network on the AMI dataset, hinting that it can be an effective tool for improved segmentation at inference time when guided by qualitative supervision.

Despite the poor performance of the Temporal Refinement Network with a smaller number of temporal input frames, the positive correlation between T and F-measure is promising as it suggests that the refinement module could very well outperform the base network when the base network gives a reasonable initial guess.

More work should be done to assess the effects network structure on Temporal Refinement Network. For example, instead of just copying the base network structure for the refinement module, another network could be used. One could also consider the effects of training the refinement module to take color images as inputs instead of probability maps, or the effects of training the base network alongside the refinement module, or train multiple base networks instead of using the same one for each temporal frame.

Additionally, work can be done to assess the affects of hyper-parameters on the Temporal Refinement Network. The refinement module was trained with 1, 3, 5, and 7 frames separated by fixed intervals of time chosen at random in the range of a quarter to five seconds. It is possible that the network could perform better with even more frames, especially given the promising results of $T = 7$, the largest value that was tested, or with a different configuration of frames and time steps. Notably, the frames were centered around the input image, and had equal time step. Neither of these constraints are strictly necessary, and could potentially hinder the training/testing of the network, though it is the author's belief that these constraints help with consistency.

It is also possible that the large variation in the time steps and/or inherent motion of the training set did not provide

sufficient training data to properly decode the motion, even with the multiple temporal sets (namely, ten) produced from each input image; the testing data was also constructed with randomly sampled time steps in the same range as the training data, which could have brought about poor performance as opposed to more consistently defined discrepancies. One could thus restrict the training/testing data so that it includes constant fixed movements (to a reasonable degree) between frames; by doing so, the refinement network can effectively make the link between movement and human presence. Note that the LIRIS and AMI datasets did not contain non-human objects in motion, such as cars or birds, so some work would have to be done to account for these features that might confuse the network.

It is also still unclear the exact circumstances that led to the Temporal Refinement Network outperforming/underperforming the base network. Although the author hypothesizes that the refinement module learned from motion from a few qualitative observations and presumptions about visual systems, this is not necessarily the case, and is best determined by further investigation.

In terms of implementation, the Temporal Refinement Network can be implemented in an offline manner by collecting and centering frames around input images. This can be readily extended to real-time data as well by maintaining a buffer of previously encountered frames and performing inference at a small time offset. One could even save the outputs of the Base Network so that multiple inferences of the same image would not be necessary, effectively cutting down on inference speed.

Overall, this paper has brought to light many of the concerns of using convolutional neural networks for the problem of skin detection, both in the spatial and temporal dimensions, yielding promising results and avenues for future exploration.

References

[1] Y. Kim, I. Hwang, and N. I. Cho. Convolutional neural networks and training strategies for skin detection. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3919–3923, Sep. 2017.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[3] Yoonsik Kim, Insung Hwang, and Nam Ik Cho. A new convolutional network-in-network structure and its applications in skin detection, semantic segmentation, and artifact reduction. *CoRR*, abs/1701.06190, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[5] C. Ma and H. Shih. Human skin segmentation using fully convolutional neural networks. In *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, pages 168–170, Oct 2018.

[6] Eman Thabet, Fatimah Khalid, Puteri Suhaiza Sulaiman, and Razali Yaakob, "Low Cost Skin Segmentation Scheme in Videos Using Two Alternative Methods for Dynamic Hand Gesture Detection Method," Advances in Multimedia, vol. 2017, Article ID 7645189, 9 pages, 2017. https://doi.org/10.1155/2017/7645189.

[7] Mohammad Reza Mahmoodi, Sayed Masoud Sayedi, and Fariba Karimi. Color-based skin segmentation in videos using a multi-step spatial method. *Multimedia Tools Appl.*, 76(7):9785–9801, April 2017.

[8] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2:1–1, 06 2016.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[12] S. L. Phung, A. Bouzerdoum, and D. Chai. Skin segmentation using color pixel classification: analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):148–154, Jan 2005.

[13] Wei Ren Tan, Chee Seng Chan, Pratheepan Yogarajah, and Joan Condell. A fusion approach for efficient human skin detection. *IEEE Transactions on Industrial Informatics*, 8(1):138–147, 2011.

[14] J. SanMiguel and S. Suja. Skin detection by dual maximization of detectors agreement for video monitoring. *Pattern Recognition Letters*, (34):2102–2109, Dec 2013.

[15] Christian Wolf, Eric Lombardi, Julien Mille, Oya Celiktutan, Mingyuan Jiu, Emre Dogan, Gonen Eren, Moez Baccouche, Emmanuel Dellandréa, Charles-Edmond Bichot, and et al. Evaluation of video activity localizations integrating quality and quantity measurements. *Comput. Vis. Image Underst.*, 127:14–30, October 2014.

[16] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, and et al. The ami meeting corpus: A pre-announcement. In *Proceedings of the Second International Conference on Machine Learning for Multimodal Interaction*, MLMI'05, page 28–39, Berlin, Heidelberg, 2005. Springer-Verlag.

[17] Suman Saha, Gurkirt Singh, Michael Sapienza, Philip H. S. Torr, and Fabio Cuzzolin. Spatio-temporal human action localisation and instance segmentation in temporally untrimmed videos. *CoRR*, abs/1707.07213, 2017.