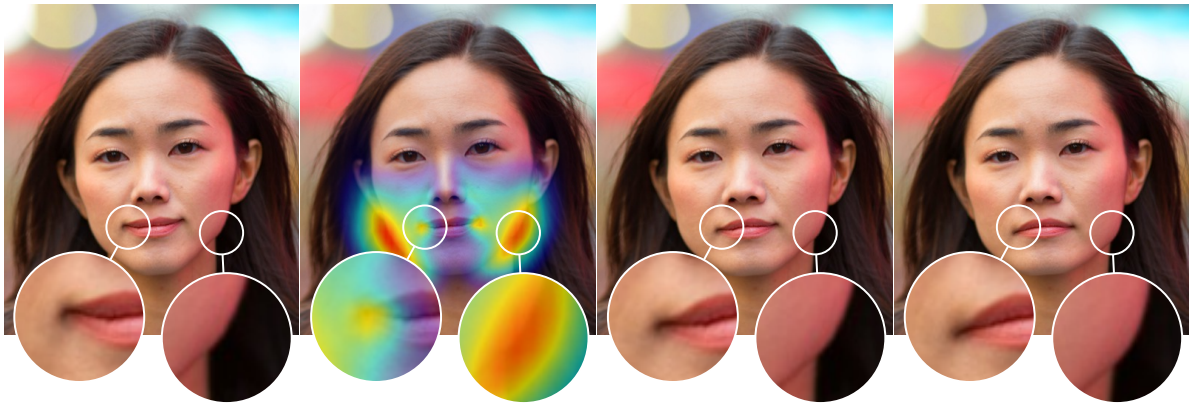
Detecting Photoshopped Faces by Scripting Photoshop

Sheng-Yu Wang[1] Oliver Wang[2] Andrew Owens[1] Richard Zhang[2] Alexei A. Efros[1]
UC Berkeley[1] Adobe Research[2]



(a) Manipulated photo (b) Detected manipulations (c) Suggested "undo" (d) Original photo

Figure 1: Given an input face (a), our tool can detect that the face has been warped with the Face-Aware Liquify tool from Photoshop, predict where the face has been warped (b), and attempt to "undo" the warp (c) and recover the original image (d).

Abstract

Most malicious photo manipulations are created using standard image editing tools, such as Adobe Photoshop. We present a method for detecting one very popular Photoshop manipulation – image warping applied to human faces – using a model trained entirely using fake images that were automatically generated by scripting Photoshop itself. We show that our model outperforms humans at the task of recognizing manipulated images, can predict the specific location of edits, and in some cases can be used to "undo" a manipulation to reconstruct the original, unedited image. We demonstrate that the system can be successfully applied to real, artist-created image manipulations.

1. Introduction

In an era when digitally edited visual content is ubiquitous, the public is justifiably eager to know whether the images they see on TV, in glossy magazines, and on the Internet are, in fact, *real*. While the popular press has mostly focused on "DeepFakes" and other GAN-based methods that may one day be able to convincingly simulate a real person's appearance, movements, and facial expressions [34, 8, 5, 16], for now, such methods are prone to degeneracies and exhibit visible artifacts [24]. Rather, it is the more subtle image manipulations, performed with classic image processing techniques, typically in Adobe Photoshop, that have been the largest contributors to the proliferation of manipulated visual content [12]. While such editing operations have helped enable creative expression, if done without the viewer's knowledge, they can have serious negative implications, ranging from body image issues set by unrealistic standards, to the consequences of "fake news" in politics.

In this work, we focus on one specific type of Photoshop manipulation – image warping applied to faces. This is an extremely common task used for "beautification" and expression editing. Face warping is an interesting problem as it is a domain that is surprisingly hard for people to detect, but it is commonly used and has wide reaching implications. We show in a user study that humans have only 53.5% accuracy in identifying such edits (chance is 50%). We propose a *lens* through which these subtle edits become visualized, alerting the viewer to the presence of modifications, as shown on Figure 1. Our proposed approach is but one tool in a larger toolbox of techniques that together, could be used to help combat the spread of misinformation, and its effects.

*The code can be found on our website.

(a) Real images (b) Manipulated images

Figure 2: Random samples from our training dataset. (a) Real images scraped from Flickr portraits (top) and Open Images [20] (bottom). (b) Random warps automatically created with Photoshop's *Face-Aware Liquify* tool. The differences are very subtle.

Our approach consists of a CNN carefully trained to detect facial warping modifications in images. As with any deep learning method, collecting enough supervised training data is always a challenge. This is especially true for forensics applications, since there are no large-scale datasets of manually created visual fakes. In this work, we solve this problem by using Photoshop itself to automatically generate realistic-looking fake training data. We first collect a large dataset of real face images, scraped from different internet sources (Figure 2a). We then directly script the *Face-Aware Liquify* tool in Photoshop, which abstracts facial manipulations into high level semantic operations, such as "increase nose width" and "decrease eye distance". By randomly sampling manipulations in this space (Figure 2b), we are left with a training set consisting of pairs of source images and realistic looking warped modifications.

We train both global classification and local warping field prediction networks on this dataset. In particular, our local prediction method uses a combination of loss functions including flow warping prediction, relative warp preservation, and a pixel-wise reconstruction loss. We present a number of applications, including a visualization overlay to draw attention to modified regions, as in Fig. 1(b), and un-warping the manipulated image to make it more like the original, as in Fig. 1(c). Finally, we evaluate our approach on a number of test cases, including images scraped from various sources, as well as with warping operations performed by other means.

2. Related work

Image forensics, or forgery detection, is an increasingly important area of research in computer vision. In this section, we focus on works that are either trained from large amounts of data, or directly address the face domain.

Face manipulation Researchers have proposed forensics methods to detect a variety of face manipulations. Zhou *et al*. [39] and Roessler *et al*. [29, 30] propose neural network models to detect *face swapping* and *face reenactment* — manipulations where one face is wholly replaced with another (perhaps taken from the same subject) after splicing, color matching, and blending. Additional work investigates detecting morphed (interpolated) faces [28] and inconsistencies in lighting from specular highlights on the eye [15]. In contrast, we consider *facial warps* which undergo subtle geometric deformations, rather than a complete replacement of the face, or the synthesis of new details.

Learning photo forensics The difficulty in obtaining labeled training data has led researchers to propose a variety of "self-supervised" image forensics approaches that are trained on automatically-generated fake images. Chen *et al*. [9] use a convolution network to detect median filtering. Zhou *et al*. [40] propose an object detection model, specifically using steganalysis features to reduce the influence of semantics. The model is pretrained on automatically created synthetic fakes using object segmentations, and subsequently fine-tuned on actual fake images. While we also generate fakes automatically, we use the tools that a typical editor would use. This enables us to detect fakes more accurately and even attempt inversion with some success. A complementary approach is exploring unsupervised forensics models that learn only from real images, without explicitly modeling the fake image creation process. For example, several models have been proposed to detect spliced images by identifying patches which come from different camera models [7, 23], by using EXIF metadata [14], or by identifying physical inconsistencies [22]. These approaches, however, are designed to detect instances of the image compositing problem, while we address a more subtle manipulation — facial structure warping.

Hand-defined manipulation cues Other image forensics work has proposed to detect fake images using hand-defined cues [12]. Early work detected resampling artifacts [27, 19] by finding periodic correlations between nearby pixels. There has also been work that detects inconsistent quantization [2], double-JPEG artifacts [6, 3], and geometric inconsistencies [25]. However, the operations performed by interactive image editing tools are often complex, and can be difficult to model. Our approach, by contrast, learns features appropriate for its task from a large dataset of manipulated images.

3. Datasets

We scrape a large dataset of real face images from the Internet, and create two datasets of fakes: a large, automatically generated set of manipulated images for training a forensics model, and a smaller set of actual manipulations done by an artist for evaluation.

Collecting real face images To obtain a diverse dataset of faces, we aggregate images from a variety of sources. First, we take all images from the Open Images dataset [20] with the "human face" label. This dataset consists of humans in-the-wild, scraped from Flickr. We also scrape Flickr specifically for portrait photography images. To isolate the faces, we use an out-of-the-box CNN-based face detector from dlib [17] and crop the face region only. All together, our face dataset contains 69k and 116k faces from OpenImages and Flickr portrait photos, respectively, of which approximately 65k are high-resolution (at least 700 pixels on the shortest side). We note that the our dataset is biased toward Flickr users, who, on average, post higher-quality photographs than users of other Internet platforms. More problematically, the Flickr user base is predominantly Western. However, as our method is entirely self-supervised, it is easy to collect and train with new data to match the test distribution for a target application.

Generating manipulated face images Our goal is to automatically create a dataset of manipulated images that, when leveraged for training, generalizes to artist-created fakes. We script the *Face-Aware Liquify (FAL)* tool [1] in Adobe Photoshop to generate a variety of face manipulations, using built-in support for JavaScript execution. We choose Photoshop, as it is one of the most popular image editing tools, and this operation, as it is a very common manipulation in portrait photography. FAL represents manipulations using 16 parameters, corresponding to higher-level semantics (*e.g.* adjusting the width of the nose, eye distance, chin height, etc.). A facial landmark detector registers a mesh to the input image, and the parameters control the mesh's vertex displacements. As shown in Figure 1, the tool can be used to make subtle, realistic manipulations, such as making a face more symmetrical. We uniformly sample the FAL parameter space. While these parameter choices are unlikely to match the changes an artist would make, we argue, and validate, that randomly sampling the space will cover the space of "realistic" operations. We modify each image from our real face dataset randomly 6 times. In all, the data we used for training is 1.295M faces – 185k unmodified, and 1.1M modified. Additionally, we hold out 5K real faces each from Open Images and Flickr, leaving half of the images unmodified and the rest modified in the same way as the training data. In total, the validation data consists of 2.5K images in each categories – {Open Images, Flickr} × {unmanipulated, manipulated}. Table 1 summarizes that data and Figure 2 shows random samples.

	Train	Val	Test
Source	OpenImage & Flickr		Flickr
Total Images	1.1M	10k	100
Unmanipulated images	157k	5k	50
Manipulated images	942k	5k	50
Manipulations	Random FAL		Pro Artist

Table 1: **Dataset statistics.** This includes our own automatically created data as well as a smaller test set of manipulations created by a professional artist.

Test Set: Artist-created face manipulations We test the generalization ability to "real" manipulations by contracting a professional artist to manipulate 50 real photographs. Half are manipulated with the intent of "beautifying", or increasing attractiveness, and the other half to change facial expression, positively or negatively. This covers two important use cases. The artist created 50 images with the FAL tool, and 50 images with the more general Liquify tool – a free-form brush used to warp images. On average, it took 7.8 minutes of editing time per image.

4. Methods

Our goal is to train a system to detect facial manipulations. We present two models: a *global classification* model, tasked with predicting whether a face has been warped, and a *local warp predictor*, which can be used to identify *where* manipulations occur, and reverse them.

4.1. Real-or-fake classification

We first address the question "has this image been manipulated?" We train a binary classifier using a ResNet-50 [13] architecture, pretrained for ImageNet classification [31], and fine-tuned to our task.

We investigate the effect of resolution by training low and high-resolution models. High-resolution models enable preservation of low-level details, potentially useful for identifying fakes, such as resampling artifacts. On the other hand, a lower-resolution model potentially contains sufficient details to identify fakes and can be trained more efficiently. We try low and high-resolution models, where the shorter side of the image is resized to 400 and 700 pixels, respectively. During training, the images are randomly flipped and cropped to 384 and 640 pixels, respectively. We experimentally find that the low-resolution model performs as strongly as high-resolution.

While we control the post-processing pipeline in our test setup, real-world use cases may contain unexpected post-processing. Forensics algorithms are often sensitive to such operations [27]. To increase robustness, we consider more aggressive data augmentation, including resizing methods (bicubic and bilinear), JPEG compression, brightness, contrast, and saturation. We experimentally find that this in-

creases robustness to perturbations at testing, even if they are not in the augmentation set.

We use the Adam optimizer [18] with $\beta_1 = 0.9, \beta_2 = 0.999$, minibatch size 64 and 14 for the low and high-res models, respectively, and initial learning rate 10^{-4}, reduced by $10\times$ when loss plateaus. The models are trained for 600k iterations on 135.4k original images and 812.4k modified images, where the original images are sampled $6\times$ more frequently to balance the class distribution.

4.2. Predicting what moved where

Upon detecting whether a face has been a modified, a natural question for a viewer is *how* the image was edited: which parts of an image were warped, and what did the image look like prior to manipulation? To do this, we predict an optical flow field $\hat{U} \in \mathbb{R}^{H\times W\times 2}$ from the original image $X_{orig} \in \mathbb{R}^{H\times W\times 3}$ to the warped image X, which we then use to try to "reverse" the manipulation and recover the original image.

We train a flow prediction model $\mathcal{F}$ to predict the per-pixel warping field, measuring its distance to an approximate "ground-truth" flow field U for each training example (computed estimating optical flow between the original and modified images). Fig. 4 shows examples of these flow fields. To remove erroneous flow values, we discard pixels that fail a forward-backward consistency test, resulting in binary mask $M \in \mathbb{R}^{H\times W\times 1}$.

$$\mathcal{L}_{epe}(\mathcal{F}) = ||M \odot \big(\mathcal{F}(X) - U\big)||_2, \tag{1}$$

where X is a manipulated image, U is its "ground-truth" flow, $\odot$ is the Hadamard product, and $\mathcal{L}_{epe}$ is a measure of flow error (also known as *endpoint error*). We compute this loss for each pixel of the image, and compute the mean. Following [33], we encourage the flow to be smooth by minimizing a multiscale loss on the flow gradients:

$$\mathcal{L}_{ms}(\mathcal{F}) = \sum_{s\in S}\sum_{t\in\{x,y\}} ||M \odot \big(\nabla_t^s(\mathcal{F}(X)) - \nabla_t^s(U)\big)||_2, \tag{2}$$

where ∇_x^s, ∇_y^s are horizontal and vertical gradients of a flow field, decimated by stride $s \in \{2, 8, 32, 64\}$.

Undoing a warp With the correct flow field predicted from the original image to the modified image, one can retrieve the original image by inverse warping. This leads to a natural reconstruction loss,

$$\mathcal{L}_{rec}(\mathcal{F}) = ||\mathcal{T}\big(X; \mathcal{F}(X)\big) - X_{orig}||_1, \tag{3}$$

where $\mathcal{T}(X; U)$ warps X by resampling with flow U. In this case, the loss is applied to the *unwarped image* directly, after warping with a differentiable bilinear interpolation layer. We note that this approach is similar to the main loss used in flow-based image synthesis models [41, 36].

Applying only the reconstruction loss leads to ambiguities in low-texture regions, which often results in undesirable artifacts. Instead, we jointly train with all three losses: $\mathcal{L}_{total} = \lambda_e\mathcal{L}_{epe} + \lambda_m\mathcal{L}_{ms} + \lambda_r\mathcal{L}_{rec}$. We find $\lambda_e = 1.5$, $\lambda_m = 15$, and $\lambda_r = 1$ work well and perform ablations in 5.2.

Architecture We use a Dilated Residual Network variant (DRN-C-26) [37], pretrained on the ImageNet [31] dataset, as our base network for local prediction. The DRN architecture was designed originally for semantic segmentation, and we found that it worked well for our warp prediction task.

We found that training for flow regression network directly performs poorly. We first recast the problem into multinomial classification, commonly used in regression problems (e.g., colorization [21, 38], surface normal prediction [35], and generative modeling [26]), and then fine-tune for regression. We computed ground truth flow fields using PWC-Net [32]. Details of the training procedure are in Appendix B.

5. Experiments

We evaluate our ability to detect and undo image manipulations, using both automatic and artist-created images.

5.1. Real-or-fake classification

We first investigate whether manipulated images can be detected by our global classifier on our validation set. We test the robustness of the classifier by perturbing the images, and measure its generalization ability to manipulations by a professional artist (Table 2).

We evaluate several variants: **(1) Our full method**: a lower-resolution model (400 pixels on the smaller side), with data augmentation (compression, resizing methods, and photometric changes) and the whole training set (including low-resolution images). **(2) No augmentation**: We test the augmentation methods above by omitting them. Note that all models still include random flipping and cropping. **(3) High-res**: We test if keeping higher resolution (700 pixels on shorter side) may allow the network to pick up on more details. We keep the lower resolution images by upsampling them. **(4) High-res, partial set**: We test if including the lower-resolution images hurts performance in the previous test and exclude them from this model during training.

Baselines We compare our approach to several recent methods, which were trained for other, related forensics tasks. **(1) FaceForensics++** [30]: A network trained on face swapping and reenactment data; we use the Xception [10] model trained on raw video frames. **(2) Self-consistency** [14]: A network trained to spot low-level inconsistencies within an image.

Algorithm				Validation (Random FAL)					Test (Professional Artist)				
Method	Resolution	with Aug?	Full set?	Accuracy			AP	2AFC	Accuracy			AP	2AFC
				Orig	Mod	Total			Orig	Mod	Total		
Chance	–	–	–	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0
Human	–	–	–	–	–	–	–	53.5	–	–	–	–	71.1
FaceForensics++ [30]	–	–	–	97.1	2.0	49.7	49.0	–	7.1	95.2	51.2	53.6	61.9
Self-consistency [14]	–	–	–	–	–	–	52.6	–	–	–	–	56.4	72.0
High-res, partial set	700	✓		87.5	78.8	83.1	93.0	97.8	98.0	70.0	**84.0**	93.6	92.0
High-res	700	✓	✓	97.7	89.5	93.6	99.1	99.0	76.0	66.0	71.0	81.6	**96.0**
No augmentation	400		✓	95.6	94.8	**95.2**	**99.4**	**99.4**	80.0	70.0	77.0	84.6	90.0
Our Full	400	✓	✓	93.8	93.9	93.9	98.9	**99.4**	90.0	78.0	**84.0**	**94.7**	92.0

Table 2: **Global classifier performance.** Our best-performing method uses 400-pixel resolution, trained with data augmentation. Though this achieves slightly lower performance than training without augmentation (98.9 vs 99.4 mAP), it is more robust to corruptions, both within and outside of the augmentation set (see Fig 3). As a result, it also performs stronger on real-world uses from an artist.

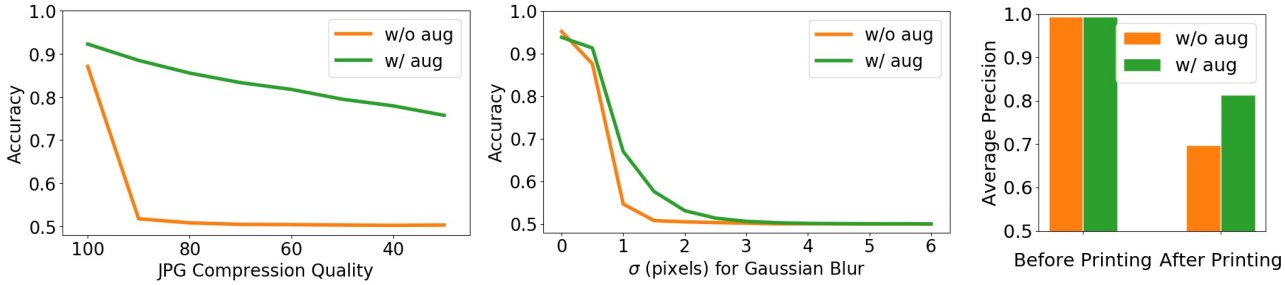


Figure 3: **Robustness to corruptions.** Accuracy of global classification with JPEG, Gaussian blur, and after rescanning, with and without data augmentation. When training with augmentation, we observed increased robustness to several perturbations. (left) JPEG compression: though unsurprising, as it is in our augmentation set, it is important, as compression and recompression is commonly applied to images. (middle) Blur: though this is not in our augmentation set, we observe increased robustness. (right) Rescanning: we corrupt by printing and rescanning a subset of photos, a perturbation that cannot be reproduced during training

Evaluations First, we evaluate its raw accuracy on the binary prediction task. Second, we use ranking-based scores that are not sensitive to the "base rate" of the fraction of fake images (which may be difficult to know in practice). For this, we use Average Precision (AP), as well as a Two Alternative Force Choice (2AFC) score that is directly comparable to human studies, where we provide our model with two images, one real and one manipulated, and measure the fraction of the time it assigns a higher manipulation probability to the fake.

Evaluation on auto-generated fakes We first explore performance on our validation set, shown in Table 2 (left), containing automatically-generated manipulated images.

We begin by running a human studies test on Amazon Mechanical Turk (AMT). We show real and manipulated images, side-by-side for 6 seconds, and ask participants to identify the one that was modified. We gave 15 example pairs to "train" each person and then collected 35 test examples (for 40 participants total). Since the manipulations we train with are subtle, this a challenging task; participants were able to identify the manipulated image 53.5% of the time (chance = 50%). This indicates that it is difficult to use high-level semantics alone for this task.

Our model, which can take advantage of both high-level and low-level cues, performs at 93.9% accuracy and 98.9% average precision. Processing at higher resolution, 700 pixels, achieves roughly the same performance in accuracy and AP, indicating that 400 pixels is large enough resolution to preserve any low-level cues the network may be using. Without augmenting for different resampling techniques, our network performance increases to 95.2% accuracy and 99.4% AP. However, doing so leaves the network susceptible to corruptions, which we explore next.

Robustness to corruptions One question is whether the network is using "high-level" cues, such as shape and symmetry, or simply relying on "low-level" cues, such as resampling artifacts. To investigate this, we perturb the validation images by changing their low-level statistics through lossy JPEG compression, blurring, and printing and scanning physical prints. This offers three interesting test cases,

	Losses			Face-Aware Liquify (FAL)						Other Manipulations					
				Val (Rand-FAL)			Artist-FAL			Artist-Liquify			Portrait-to-Life [4]		
	EPE	Multi-scale	Pix ℓ_1	EPE ↓	IOU-3 ↑	ΔPSNR ↑	EPE ↓	IOU-3 ↑	ΔPSNR ↑	EPE ↓	IOU-3 ↑	ΔPSNR ↑	EPE ↓	IOU-3 ↑	ΔPSNR ↑
EPE-only	✓			**0.51**	0.44	+2.51	0.73	0.30	+2.03	**0.58**	**0.08**	-0.96	**1.77**	0.38	–
MultiG	✓	✓		0.55	0.42	+2.23	0.77	**0.31**	+2.17	0.61	**0.08**	-0.94	1.78	0.41	–
Full	✓	✓	✓	0.52	**0.50**	**+4.41**	**0.70**	**0.31**	**+2.55**	0.61	**0.08**	**-0.61**	1.78	**0.43**	–

Table 3: **Local network performance.** We show performance of our local prediction models across several evaluations: (1) EPE, which measures average flow accuracy, (2) IOU-3, which measures flow magnitude prediction accuracy and (3) ΔPSNR, which measures how closely the predicted unwarping recovers the original image from the manipulated; ↑, ↓ indicate if higher or lower is better. Our full method with all losses (flow prediction, multiscale flow gradient, and pixel-wise reconstruction) performs more strongly than ablations, both across datasets which use Face-Aware Liquify and other manipulations.

as we *did* train on JPEG compressed images, did *not* train on blurring, and *cannot* train on rescanned images.

As shown in Fig. 3, the method with augmentation is fairly robust to JPEG compression. Though we did not train with blurring augmentations (as images are unlikely to be intentionally blurred), training with other augmentations helps increase resilience. This indicates that although the network may be taking advantage of certain low-level cues, such as resampling, some performance is retained even when those are averaged away. However, we can see that the classifier is relying on some high frequency information, since with significant blur ($\sigma > 4$), performance degrades to chance levels.

Lastly, we also test the robustness of our classifier to print rebroadcasting [11], testing on images that are printed, and then re-digitized by a scanner (*e.g.*, simulating the task of identifying manipulations in magazine covers). We used a Canon imageRunner Advance C3530i Multifunctional copier and standard 8.5×11 inch paper. We randomly selected 30 images each from the Flickr and OpenImages sets. Classification performance drops from 95.8% to 70.0% (standard error of 5.9%). While rebroadcasting hurts performance, our model still detects manipulated images significantly more accurately than chance.

Artist test set Critically, we investigate if training on our random perturbations generalizes to a more real-world setting. We collect data from a professional artist, tasked with the goal of making a subject more attractive, or changing the subject's expression. As the edits here are made to be more noticeable, and study participants were able to identify the modified image with 71.1% accuracy. Our classifier achieves 92.0% in the 2AFC setting. Our accuracy drops from 93.9 in the validation setting to 84.0. However, the AP drops much less, from 98.9 to 94.7. This indicates that there is some domain gap between our random perturbations and an artist, which can be reduced by a certain extent by "recalibrating" the classifier's detection threshold.

Baselines We compare our method to two recent baselines for image forensics, FaceForensics++ [30] and Self-consistency [14]. Importantly, neither of these methods are designed for our use case: FaceForensics++ is split into three manipulation types: face swapping, "deepfakes" face replacement, and face2face reenactment [30]. Self-consistency, on the other hand, is designed to detect low-level differences in image characteristics. Both methods perform around chance on our dataset, indicating that generalizing to facial warping manipulations is a challenging task.

However, our method seems to generalize to some of the FaceForensics++ datasets. Our full method achieves above chance on FaceSwap (64.7% AP) and DeepFake (70.9% AP), but chance on Face2Face (49.0% AP). This indicates that training on subtle facial warping data can possibly generalize to other, more complex, editing tasks.

5.2. Localizing and undoing manipulations

Next, we evaluate manipulation localization and reversal.

Model variations To help understand what parts of our model contributed to its performance, we ablate the loss functions for our local prediction model. Since previous methods have not considered the problem of predicting or reversing warps, we consider variations of our own model. **(1) Our full method**: trained with endpoint error (EPE) (Eqn. 1), multiscale gradient (Eqn. 2), and reconstruction (Eqn. 3) losses. **(2) EPE**: an ablation only trained with endpoint loss. **(3) MultiG**: trained with endpoint and multiscale, but without reconstruction loss.

Evaluations We evaluate our model in several ways, capturing both localization ability and warp reversal ability. **(1) End Point Error (EPE)** similarity between predicted and ground-truth flow between the original and manipulated images (Eqn 1). **(2) Intersection Over Union (IOU-τ)** We apply threshold τ to predicted and ground truth flows magnitudes and compute IOU. **(3) Delta Peak**

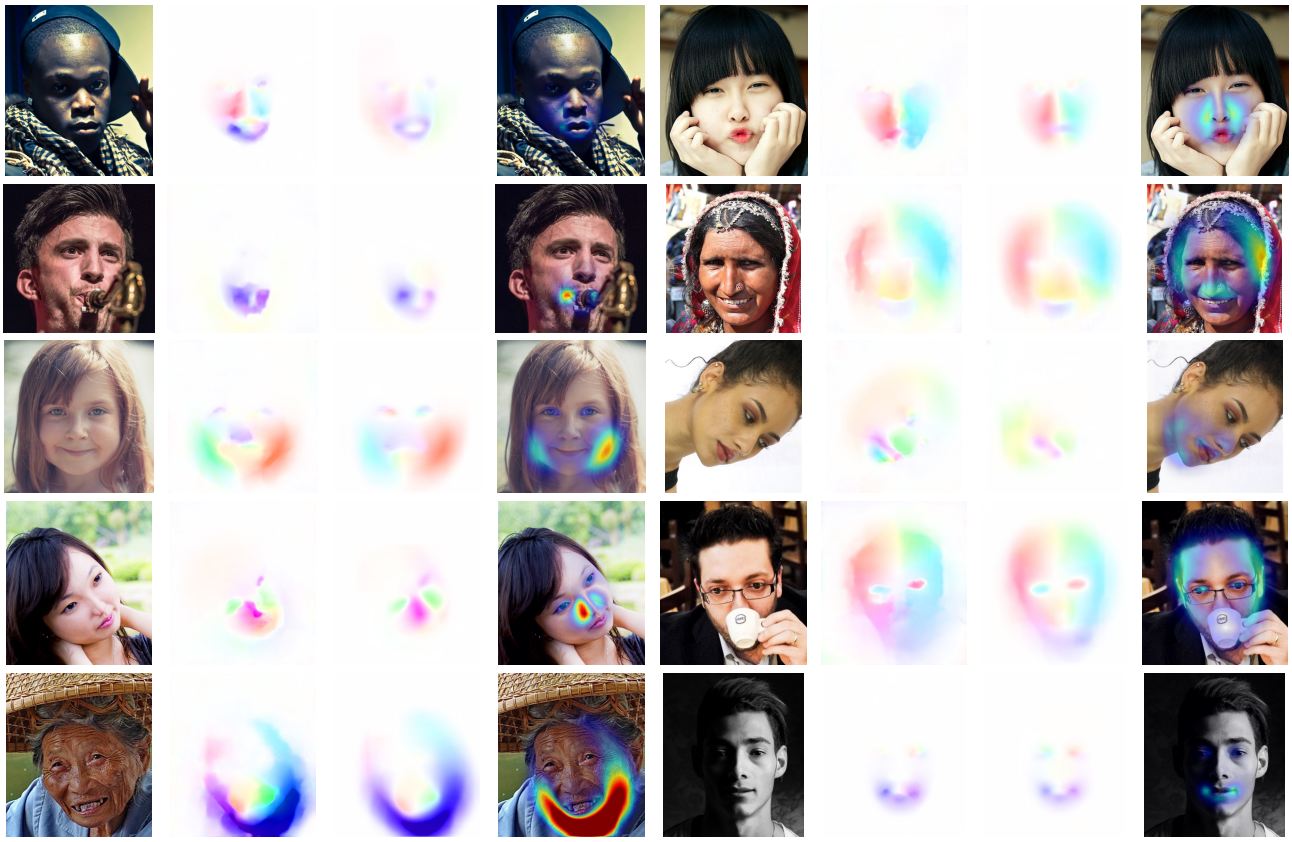


Figure 4: **Qualitative results on artist data and validation set.** We show examples of our flow prediction on images manipulated from an external artist and from our validation set. **(Input)** Input manipulated image. **(GT flow)** The "ground truth" optical flow from original to manipulated image. **(Our prediction)** Predicted flow from our network. **(Flow overlay)** Magnitude of predicted flow overlaid. See the appendix for additional examples.

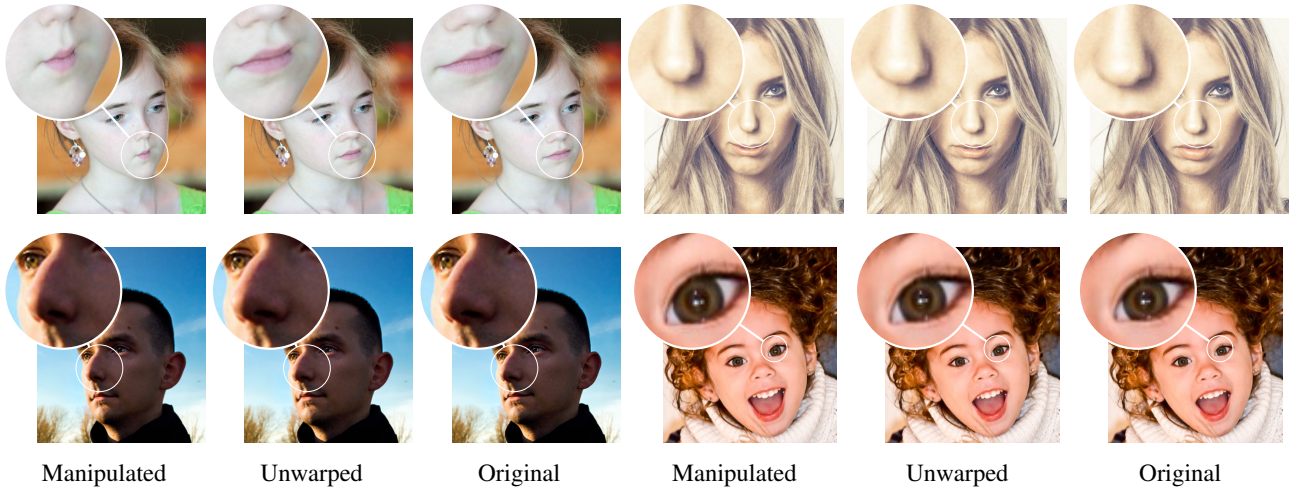


Figure 5: **Unwarping results.** These images show results from the artist edited test dataset, where the subtle manipulations are reversed. Among other edits, the mouth and nose in the top row were expanded, and the nose shape was made less round and the eye was shrunk in the bottom row.

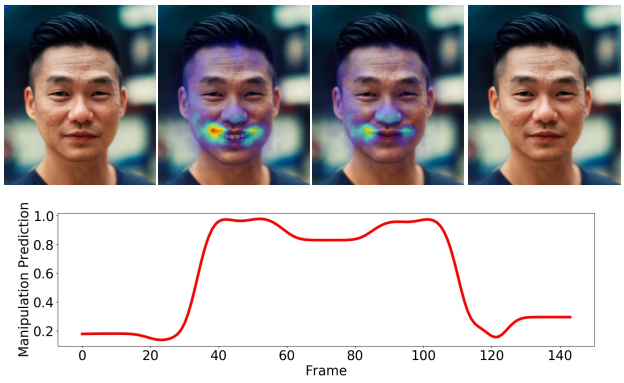


Figure 6: **Analysis of a Puppeteered video [4]**. A single input image (first and last frame) is driven by a smiling animation. **(top)** Our local analysis correctly identifies manipulated regions (corners of the mouth). **(bottom)** The global prediction over time shows how the animation moves from input to smile and back.

Signal-to-Noise Ratio (ΔPSNR) effectiveness of our predicted unwarping, PSNR(original, unwarped manipulated) minus PSNR(original, manipulated)

Analysis As shown in Table 3, we found that removing a loss significantly reduced performance. In particular, we found that directly optimizing the reconstruction loss led to significantly better image reconstruction. In Figures 4 and 5, we show several qualitative results on the automatically-generated and artist-created data. We include more qualitative results, randomly sampled from the validation set, in the appendix.

5.3. Out-of-distribution manipulations

Although the goal of our model is to detect face warping manipulations made by Photoshop, we also evaluate its ability to detect other kinds of image editing, and discuss its limitations.

Puppeteering We conduct an experiment to see whether our method can be used to detect the results of recent image-puppeteering work [4]. In this work, a video (from a different subject) is used to animate an input image via image warping and the additional of extra details, such as skin wrinkles and texture in the eyes and mouth. We apply our manipulation detection model to this data, and show that despite not being trained on this data, we are still able to make reasonable predictions. Fig 6 shows a qualitative result of running both local and global predictors on this data, where it correctly identifies a puppeted smile animation that starts and returns to a (real) rest pose. We note that PSNR comparisons on this data are not possible, due to the addition of non-warping image details.

Generic Liquify filter Like any data-driven method, we are limited by our training distribution. Warping edits that exist outside of this, such as warping applied to hair or body, cannot be detected by our method. This can be seen in our artist experiment with the generic (non-face) Liquify filter, where images are sometimes outside the distribution (Figure 7). Despite this, our method can still predict with success well above chance (63.0 accuracy, 75.8 AP), indicating some generalization. However, the global classifier performs well below the FAL operation (84.0 accuracy, 94.7 AP), and the local prediction accuracy is not enough to improve the PSNR when unwarping (-0.61 ΔPSNR). Increasing the range of scripted warping operations is likely to improve this. In general, reversing the warps is a challenging problem, as there are many configurations of plausible faces. This can be seen in that the PSNR improvement we get on the artist test set is limited to +2.55 db on average. While this manipulation is reduced, the problem of perfectly restoring the original image remains an open challenge.

GT edits Predicted edits Unwarped diff

Figure 7: **Limitations**. When manipulations are too far outside the training distribution, as with the general Liquify tool experiment. Our local prediction model fails to correctly identify warped regions. This is visible in the overlay as well as in the unwarped image (difference to ground truth after unwarping is shown on the right, darker is worse).

6. Conclusion

We have presented the first method designed to detect facial warping manipulations, and did so using by training a forensics model entirely with images automatically generated from an image editing tool. We showed that our model can outperform human judgments in determining whether images are manipulated, and in many cases is able to predict the local deformation field used to generate the warped images. We see facial warp detection as an important step toward making forensics methods for analyzing images of a human body, and extending these approaches to body manipulations and photometric edits such as skin smoothing are interesting avenues for future work. Moreover, we also see our work as being a step toward toward making forensics tools that learn without labeled data, and which incorporate interactive editing tools into the training process.

7. Acknowledgements

We thank Daichi Ito and Adam Pintek for contributing to our artist test set, and Jacob Huh for the helpful discussions. This work was supported, in part, by DARPA MediFor and UC Berkeley Center for Long-Term Cybersecurity. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] Adjust and exaggerate facial features. https://helpx.adobe.com/photoshop/how-to/face-aware-liquify.html. 3

[2] S. Agarwal and H. Farid. Photo forensics from jpeg dimples. In *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2017. 2

[3] I. Amerini, T. Uricchio, L. Ballan, and R. Caldelli. Localization of jpeg double compression through multi-domain convolutional neural networks. In *Proc. of IEEE CVPR Workshop on Media Forensics*, 2017. 2

[4] H. Averbuch-Elor, D. Cohen-Or, J. Kopf, and M. F. Cohen. Bringing portraits to life. *ACM Transactions on Graphics (TOG)*, 36(6):196, 2017. 6, 8

[5] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh. Recycle-gan: Unsupervised video retargeting. In *ECCV*, 2018. 1

[6] M. Barni, L. Bondi, N. Bonettini, P. Bestagini, A. Costanzo, M. Maggini, B. Tondi, and S. Tubaro. Aligned and non-aligned double jpeg detection using convolutional neural networks. *Journal of Visual Communication and Image Representation*, 49:153–163, 2017. 2

[7] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro. Tampering detection and localization through clustering of camera-based cnn features. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1855–1864. IEEE, 2017. 2

[8] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros. Everybody dance now. *arXiv preprint arXiv:1808.07371*, 2018. 1

[9] J. Chen, X. Kang, Y. Liu, and Z. J. Wang. Median filtering forensics based on convolutional neural networks. *IEEE Signal Processing Letters*, 22(11):1849–1853, 2015. 2

[10] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 4

[11] W. Fan, S. Agarwal, and H. Farid. Rebroadcast attacks: Defenses, reattacks, and redefenses. In *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018. 6

[12] H. Farid. *Photo forensics*. MIT Press, 2016. 1, 2

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[14] M. Huh, A. Liu, A. Owens, and A. A. Efros. Fighting fake news: Image splice detection via learned self-consistency. In *ECCV*, 2018. 2, 4, 5, 6

[15] M. K. Johnson and H. Farid. Exposing digital forgeries through specular highlights on the eye. In *International Workshop on Information Hiding*, pages 311–325. Springer, 2007. 2

[16] H. Kim, P. Carrido, A. Tewari, W. Xu, J. Thies, M. Niessner, P. Pérez, C. Richardt, M. Zollhöfer, and C. Theobalt. Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4):163, 2018. 1

[17] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10(Jul):1755–1758, 2009. 3

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4, 10

[19] M. Kirchner. Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue. In *Proceedings of the 10th ACM workshop on Multimedia and security*, pages 11–20. ACM, 2008. 2

[20] I. Krasin, T. Duerig, N. Alldrin, A. Veit, S. Abu-El-Haija, S. Belongie, D. Cai, Z. Feng, V. Ferrari, V. Gomes, et al. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github. com/openimages*, 2(6):7, 2016. 2, 3

[21] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593. Springer, 2016. 4

[22] J. Li, X. Li, B. Yang, and X. Sun. Segmentation-based image copy-move forgery detection scheme. *IEEE Transactions on Information Forensics and Security*, 10(3):507–518, 2015. 2

[23] O. Mayer and M. C. Stamm. Learned forensic source similarity for unknown camera models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2012–2016. IEEE, 2018. 2

[24] K. McDonald. How to recognize fake ai-generated images, December 2018. https://medium.com/@kcimc/how-to-recognize-fake-ai-generated-images-4d1f6f9a2842. 1

[25] J. F. O'Brien and H. Farid. Exposing photo manipulation with inconsistent reflections. *ACM Trans. Graph.*, 31(1):4–1, 2012. 2

[26] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016. 4

[27] A. C. Popescu and H. Farid. Exposing digital forgeries by detecting traces of resampling. *IEEE Transactions on signal processing*, 53(2):758–767, 2005. 2, 3

[28] R. Raghavendra, K. B. Raja, S. Venkatesh, and C. Busch. Transferable deep-cnn features for detecting digital and print-scanned morphed face images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1822–1830. IEEE, 2017. 2

[29] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. FaceForensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018. 2

[30] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. FaceForensics++: Learning to detect manipulated facial images. *arXiv preprint arXiv:1901.08971*, 2019. 2, 4, 5, 6

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 3, 4

[32] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 4, 10

[33] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *CVPR*, 2017. 4

[34] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 1

[35] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–547, 2015. 4

[36] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, pages 1–20. 4

[37] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

[38] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 4, 10

[39] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Two-stream neural networks for tampered face detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE*, pages 1831–1839, 2017. 2

[40] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Learning rich features for image manipulation detection. In *CVPR*, 2018. 2

[41] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *ECCV*, 2016. 4

Appendix

A. Qualitative results

Dataset Figure 9 shows a sample of the manipulations in our automatically-generated dataset. For each example photo, we show all 6 random manipulations that were applied to it.

Local predictions Figure 10 shows a random selection of results from our validation dataset of automatically-generated manipulations. Figure 8 shows the psnr change with respect to a multiplicative factor on the predicted flow field over our validation dataset.

B. Implementation details for warp prediction

Flow consistency mask Given the original image X_{orig} and manipulated image X_{mod}, we compute the flow from original to manipulated and from manipulated to original using PWC-Net [32], which we denote U_{om} and U_{mo}, respectively.

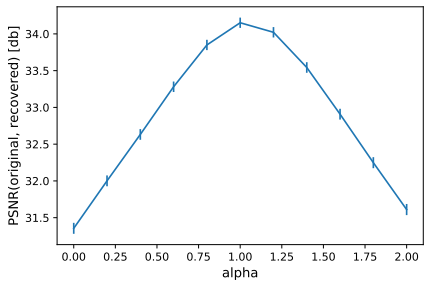


Figure 8: PSNR plot from our held-out validation subset, where the y-axis is average PSNR of the unwarped image to the original, and the x-axis is multiplicative factor on the predicted flow field. The error bars are the standard errors. In the ideal case, this PSNR should peak at 1.0, the predicted flow.

To compute the flow consistency mask, we transform U_{mo} from the manipulated image space into the original image space, which is $U'_{mo} = \mathcal{T}(U_{mo}; U_{om})$. We consider the flow to be consistent at a pixel if the magnitude of $U'_{mo} + U_{om}$ is less than a threshold. After this test, pixels corresponding to occlusions and ambiguities (*e.g.*, in low-texture regions) will be marked as inconsistent, and therefore do not contribute to the loss.

We take relative error of the flow consistency as the criterion. For a pixel p,

$$M_{inconsistent}(p) = \mathbb{1}\Big\{\frac{||U'_{mo}(p) + U_{om}(p)||_2}{||U_{om}(p)||_2 + \epsilon} > \tau\Big\}. \quad (4)$$

We take $\epsilon = 0.1$ and $\tau = 0.85$, then apply a Gaussian blur with $\sigma = 7$, denoted by G, and take the complement to get the flow consistency mask $M = 1 - G(M_{inconsistent})$.

Training details for local prediction networks We use a two-stage training curriculum, where we first train a per-pixel 121-class classifier to predict the discretized warping field. We round the flow values into the closest integer, and assign class to each integer (u, v) value with a cutoff at 5 pixels. Therefore, we have $u, v \in \{-5, -4, \ldots, 4, 5\}$, *i.e.* 121 classes in total. We pretrained the model for 100k iterations with batch size 16. Our strategy is consistent to Zhang et al. [38], which found that (in the context of colorization) pretraining with multinomial classification and then fine-tuning for regression gave better performance than just training for regression directly.

The base-network of the regression model is initialized with the pretrained model weights, and the other weights are initialized with normal distribution with gain 0.02. We train the models for 250k iterations with batch size 32.

Both models are trained with Adam optimizer [18] with learning rate 10^{-4}, $\beta_1 = 0.9, \beta_2 = 0.999$.

Figure 9: A random sample of manipulations from our dataset. For each photo, we show all 6 random edits that we made. We note that many of these modifications are subtle.

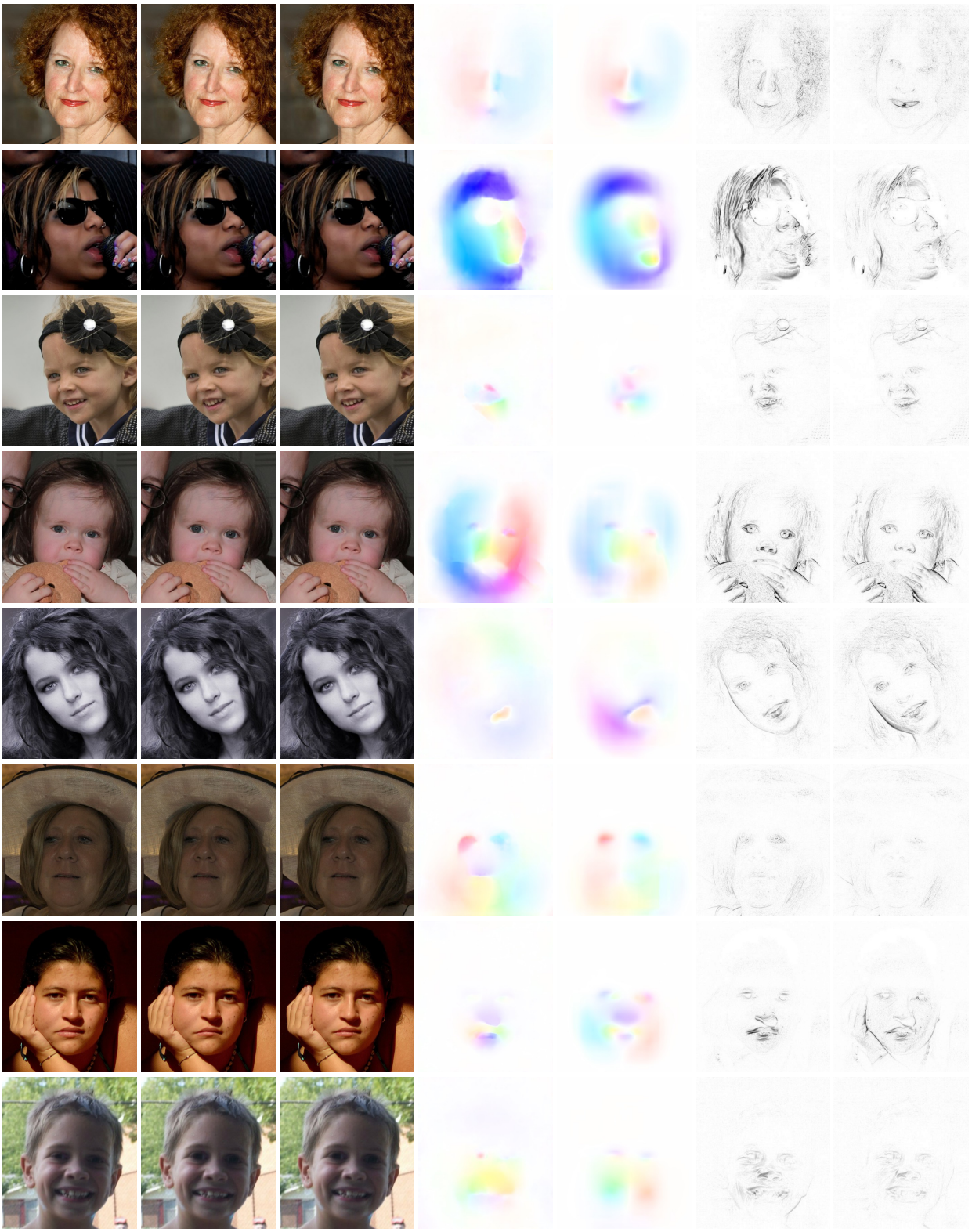


Figure 10: Randomly selected results from our held-out validation dataset, showing the original, warped, and unwarped images. The ground-truth and predicted flow fields, and the difference images between the manipulated and original image, and the unwarped and original images (enhanced for visibility).