Supplementary Material for:
Video Prediction with Appearance and Motion Conditions

Yunseok Jang [1 2] **Gunhee Kim** [2] **Yale Song** [3]

A. Architecture Details (Section 3.2)

We provide architecture details of our AMC-GAN introduced in the main paper (Section 3.2).

A.1. Generator Network (Figure A)

It takes a random noise vector $\mathbf{z} \in \mathbb{R}^p$ sampled from a normal distribution $\mathcal{N}(0, I)$, and the two conditioning variables $\mathbf{y}_a$ and $\mathbf{y}_m$ as input; we set $p = 96$ for MUG and 128 for NATOPS. The output is a video $\hat{\mathbf{x}}|_{\mathbf{y}}$, generated frame-by-frame by unrolling a convolutional LSTM (convLSTM) (Shi et al., 2015) and an image decoder network $T - 1$ times.

We encode $\mathbf{y}_a$ using five convolutional layers: `conv2d(32)` – `leakyReLU` – `conv2d(64)` – `BN` – `leakyReLU` – `pool` – `conv2d(128)` – `BN` – `leakyReLU` – `pool` – `conv2d(256)` – `BN` – `leakyReLU` – `pool` – `conv2d(256)` – `BN` – `leakyReLU`, where `conv2d(`k`)` is a 2D convolutional layer with k filters of 3×3 kernel with stride 1, `pool` is average pooling on 2×2 region with stride 2, and `BN` is batch normalization (Ioffe & Szegedy, 2015). The output is an embedding $\phi(\mathbf{y}_a)$ of size $8 \times 8 \times 256$.

We unroll the convLSTM (Shi et al., 2015) for $T - 1$ time steps to produce the output video. The convLSTM has 256 filters of 3×3 kernel with stride 1. We initialize its states using $\phi(\mathbf{y}_a)$.

At each t-th time step, we pass the motion condition $\mathbf{y}_{m,t} \in \mathbb{R}^q$ to a fully connected layer with a gated operation. That is, we compute the gate value $t = sigmoid(fc(q)) \in \mathbb{R}^1$ and obtain $t * fc(q) + (1 - t) * q$. Then, we spatially tile it to form a $8 \times 8 \times q$ tensor, which is the input to the convLSTM. In our experiments, $q = 28$ for the MUG dataset (by concatenating 11 of 2D facial landmarks and an one-hot vector of 6 emotion class) and $q = 42$ for the NATOPS dataset (by concatenating 9 of 2D body joints and an one-hot vector of 24 motion class).

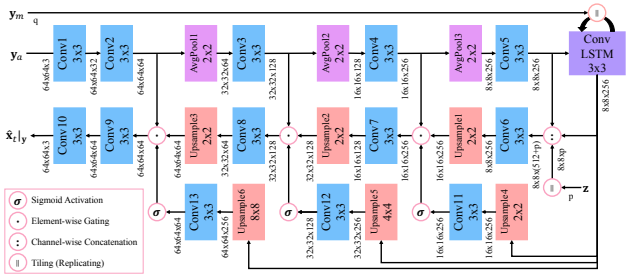

Figure A. Generator $\mathcal{G}$ network architecture (used in Figure 2 of the main paper).

We add a skip connection to create a direct path from $\phi(\mathbf{y}_a)$ to output of the convLSTM via channel-wise concatenation. We then apply the spatial tiling to the random noise vector $\mathbf{z} \in \mathbb{R}^p$ for 8×8 times (depicted as "tiling" in Figure A) and concatenate it with the other two tensors (depicted as "channel-wise concatenation" in Figure A). This makes the output of the convLSTM a $8 \times 8 \times (512 + p)$ tensor at each time step; we set $p = 96$ for the MUG dataset and 128 for the NATOPS dataset.

The image decoder (the bottom two rows in Figure A) takes the concatenated output and produces the next frame by a series of deconvolutions. To avoid the checkerboard artifact in deconvolution (Odena et al., 2016), we use the upscale-convolution trick for all deconvolutional steps. The decoder architecture is `conv2d(256)` – `BN` – `leakyReLU` – `upsample` – `gating` – `conv2d(128)` – `BN` – `leakyReLU` – `upsample` – `gating` – `conv2d(64)` – `BN` – `leakyReLU` – `upsample` – `gating` – `conv2d(64)` – `BN` – `leakyReLU` – `conv2d(3)` – `tanh`, where `conv2d(`k`)` is a 2D convolutional layer with k filters of 3×3 kernel with stride 1, and `upsample` is the 2×2 bilinear up-sampling.

To provide a skip-connection from the image encoder to the decoder, we incorporate a `gating` operator that computes a weighted average of two tensors, whose weights are computed from the output of convLSTM at each time step. Specifically, we encode the convL-

[1]University of Michigan, Ann Arbor [2]Seoul National University [3]Microsoft AI & Research. Correspondence to: Yunseok Jang <yunseokj@umich.edu>, Gunhee Kim <gunhee@snu.ac.kr>, Yale Song <yalesong@microsoft.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

STM output with a small network of `upsample(2) – conv2d(256) – leakyReLU – sigmoid` for `[Conv11]`, `upsample(4) – conv2d(128) – leakyReLU – sigmoid` for `[Conv12]` and `upsample(8) – conv2d(64) – leakyReLU – sigmoid` for `[Conv13]`, where `upsample(`k`)` is a bilinear up-sampling of $k \times k$ kernel. The output of these small networks are used as weights for the gates. We then perform a weighted average of two tensors element-wise (depicted as "element-wise gating" in Figure A). Formally, denoting the output of the small network (e.g., sigmoid output of `[Conv11]`) by s, the result of the 2×2 upsampling (e.g., output of `[Upsample1]`) by u, and the tensor from the encoder via skip connection (e.g., output of `[Conv4]`) by e, the element-wise gating computes: $s \cdot u + (1 - s) \cdot e$, where $\cdot$ is element-wise multiplication.

A.2. Appearance Discriminator Network (Figure B)

Our appearance discriminator takes four frame images as input: an appearance condition $\mathbf{y}_a$ (i.e., the first frame of a video) and three consecutive frames $\mathbf{x}_{t-1:t+1}$ from either a real or a generated video. It then outputs a scalar value indicating whether the quadruplet input is real or fake.

We feed each image into a network of `conv2d(64,2) – BN – leakyReLU – conv2d(128,2) – BN – leakyReLU – conv2d(256,2) – BN – leakyReLU`, and concatenate the output from $\mathbf{y}_a$ and $\mathbf{x}_{t-1:t+1}$ channel-wise. We then take `deconv2d(256) – BN – leakyReLU – conv2d(512,2) – BN – leakyReLU – conv2d(1024,4) – BN – leakyReLU – fc(64) – BN – leakyReLU –` σ`(fc(1))`, where `conv2d(`k, s`)` is a 2D convolutional layer with k filters of 4×4 kernel with stride s, `deconv2d(`k`)` is a 2D deconvolutional layer with k filters of 3×3 kernel with stride 1 and `fc(`k`)` is a fully-connected layer with k units.

A.3. Motion Discriminator Network (Figure C)

This network takes a video $\mathbf{x}$ with *matched* appearance condition $\mathbf{y}_a$, and a motion class category $\mathbf{y}_m^l$ as input. It predicts three variables: a scalar indicating whether the video is real or fake, $\hat{y}_m^l \in \mathbb{R}^c$ representing motion categories, and $\hat{y}_m^v \in \mathbb{R}^{2k}$ representing the velocity of k keypoints.

We encode each frame of $\mathbf{x}$ and $\mathbf{y}_a$ with `conv2d(64) – leakyReLU – conv2d(128) – BN – leakyReLU – conv2d(256) – BN – leakyReLU`, where `conv2d(`k`)` is a 2D convolutional layer with k filters of 4×4 kernel with stride of 2. We then use the encoded $\mathbf{y}_a$ to initialize the hidden states of the convLSTM, which has 256 filters of 3×3 kernel with stride 1. At each time step t, we feed the encoded frame $\mathbf{x}_t$ to the convLSTM to produce output $\mathbf{o}_t$.

In each time step output $\mathbf{o}_t$, we feed it to `conv2d(64,2) – BN – leakyReLU – flatten – tanh(fc(#points x 2))` to predict the velocity at each time step, where `conv2d(`k, s`)` is a 2D convolutional layer with k filters of 4×4 kernel with stride of s, `fc(`k`)` is a fully-connected layer with k units. Similarly, with the last hidden state $\mathbf{h}_{T-1}$, we feed it to `conv2d(64,2) – BN – leakyReLU – flatten – fc(64) – BN – leakyReLU – fc(#class) – BN – leakyReLU – softmax` to predict the motion class category. Also, for conditional prediction, we share the output of `conv2d(64,2) – BN – leakyReLU` step and concatenate the motion class after replicating them. After that, we feed it to `conv2d(64,4) – BN – leakyReLU –` σ`(fc(1))` to judge whether given video have matched motion or not.

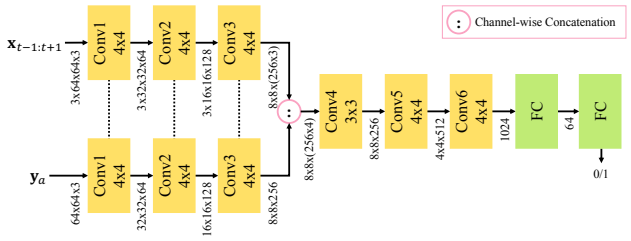

Figure B. Appearance discriminator $\mathcal{D}_a$ network architecture.

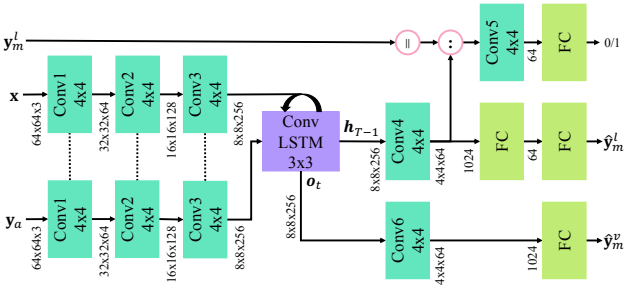

Figure C. Motion discriminator $\mathcal{D}_m$ network architecture. Dashed lines indicate parameter sharing.

B. Experiment Details

B.1. Datasets

MUG facial expression (Aifanti et al., 2010): The dataset contains 931 video clips performing six basic emotions (Ekman, 1992) (anger, disgust, fear, happy, sad, surprise). We preprocess it so that each video has 32 frames with 64×64 pixels (see below for details). For data augmentation we perform random horizontal flipping, and we use a random stride (1, 2, or 3) to sample frames around the "peak" frames. This results in 3,840 video clips, where we use 472 videos as test data.

For preprocessing, we use the OpenFace toolkit (Baltrušaitis et al., 2016) to detect facial landmarks and action unit (AU)

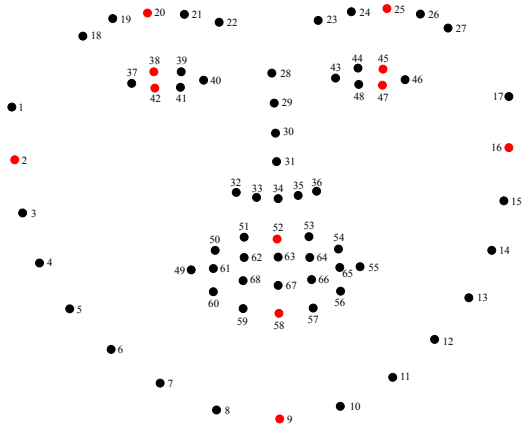

Figure D. The 68 facial-landmark template used by OpenFace (Baltrušaitis et al., 2016). We used 11 landmarks as facial keypoints (highlighted in red).

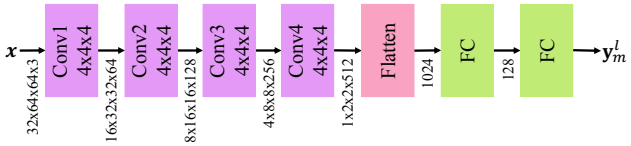

Figure E. Image-based Motion predictor network architecture

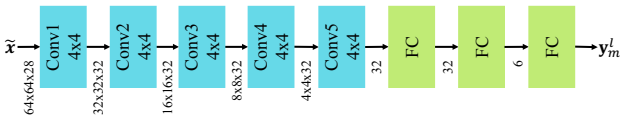

Figure F. Keypoint-heatmap based motion predictor network architecture

intensities (see Figure D). We consider only those AUs that are part of the EMFACS prototypes (Friesen & Ekman, 1983), listed in Table A, under each video's ground truth emotion category. We identify one peak frame from each video that contains the maximum AU intensity (regardless of AU) and sample 32 frames around it (23 frames before and 8 after). Next, we use facial landmarks to center-align, rescale, and crop face regions to 64×64 pixels. We use 11 facial landmarks (2, 9, 16, 20, 25, 38, 42, 45, 47, 52, 58th) as the keypoints (shown in Figure D).

NATOPS human action (Song et al., 2011): The dataset consists of 9,600 video clips performing 24 action categories. We discard 765 clips that contain less than 32 frames, resulting in 8,835 clips; we use 1,810 clips as test data. We crop the video to 180×180 pixels with the chest at the center position and rescale it to 64×64 pixels. We use 9 joint locations (head, chest, naval, L/R-shoulders, L/R-elbows, L/R-wrists) available in the dataset as keypoints.

B.2. The Loss Weights for Different Loss Terms

Table B summarizes the loss weights for different loss terms in our model that we used for each dataset.

B.3. 3D CNN Motion Classifier (Figure E)

We design a c-way motion classifier using a 3D CNN (Tran et al., 2015) that predicts the motion label $\mathbf{y}_m^l$ from a video. This network takes a video $\mathbf{x}$ as input and predicts a motion category label $\mathbf{y}_m^l$. To prevent the classifier from predicting the label simply by seeing the input frame(s), we discard the first four frames from the generated videos and use only the last 28 generated frames as input; we pad the first and last frame twice, respectively, and feed the 32 frames as input to the classifier.

We use a 3D CNN architecture of `conv3d(64)` – `leakyReLU` – `conv3d(128)` – `BN` – `leakyReLU` – `conv3d(256)` – `BN` – `leakyReLU` – `conv3d(512)`, where `BN` is batch normalization (Ioffe & Szegedy, 2015) and `conv3d(`k`)` is a 3D convolutional layer (Tran et al., 2015) with k filters of $4 \times 4 \times 4$ kernel. We use stride 2 for the first three `conv3d` layers and stride 4 for the last one. The output is an embedding $\phi(\mathbf{x})$ of size $2 \times 2 \times 512$. We flatten this network to the size of 2048 vectors and then feed it into `fc(128)` – `BN` – `leakyReLU` – `dropout(0.5)` – `fc(`c`)` – `softmax`. We set $c = 6$ for the MUG facial expression dataset and $c = 24$ for the NATOPS human action dataset.

B.4. Keypoint-based Motion Predictor (Figure F)

This network takes a series of keypoint heatmaps $\tilde{\mathbf{x}}$, obtained from a video $\mathbf{x}$ as input and predicts the motion class category $\mathbf{y}_m^l$. Similar to the image-based motion predictor, we discard the first four frames from generated videos in order to avoid the video classifier learning to categorize

Base Emotions	EMFACS Prototypes
Disgust	9
	$9 + 16 + 25, 26$
	$9 + 17$
	10^*
	$10^* + 16 + 25, 26$
	$10 + 17$
Surprise	$1 + 2 + 5B + 26, 27$
	$1 + 2 + 5B$
	$1 + 2 + 26, 27$
	$5B + 26, 27$
Anger	$(4 + 5^* + 7 + 10^* + 22 + 23 + 25, 26)^{**}$
	$(4 + 5^* + 7 + 10^* + 23 + 25, 26)^{**}$
	$(4 + 5^* + 7 + 23 + 25, 26)^{**}$
	$(4 + 5^* + 7 + 17 + 23, 24)^{**}$
	$(4 + 5^* + 7 + 23, 24)^{**}$
Happiness	$6 + 12^*$
	$12C/D$
Sadness	$(1 + 4 + 11 + 15B + /-54 + 64) + /-25, 26$
	$(1 + 4 + 15^* + /-54 + 64) + /-25, 26$
	$(6 + 15^* + /-54 + 64) + /-25, 26$
	$(1 + 4 + 15B + /-54 + 64) + /-25, 26$
	$(1 + 4 + 15B + 17 + /-54 + 64) + /-25, 26$
	$(11 + 15B + /-54 + 64) + /-25, 26$
	$11 + 17 + /-25, 26$
Fear	$1 + 2 + 4 + 5^* + 20^* + 25, 26,$ *or* 27
	$1 + 2 + 4 + 5^* + 25, 26,$ *or* 27
	$1 + 2 + 4 + 5^* + L$ *or* $R20^* + 25, 26,$ *or* 27
	$1 + 2 + 4 + 5^*$
	$1 + 2 + 5Z, +/-25, 26, 27$
	$5^* + 20^* + /-25, 26, 27$

Table A. The EMFACS (emotional facial action coding system) prototype table (Fasel et al., 2004) that we used to select relevant action units (Section 4.1). * In this combination the AU may be at any level of intensity. ** Any of the prototypes can occur without any one of the following AUs: 4, 5, 7, or 10.

Loss term	MUG	NATOPS
$\mathcal{L}_{gan}$	3.0	1.0
$\mathcal{L}_{rank}$	100.0	100.0
$\mathcal{L}_{CE}$	0.3	0.03
$\mathcal{L}_{MSE}$	300.0	30.0
$\|\mathbf{x}\|_{\mathbf{y}} - \hat{\mathbf{x}}\|_{\mathbf{y}}\|_1$	0.1	3.0
$\sum_j d_j(\mathbf{x}\|_{\mathbf{y}}, \hat{\mathbf{x}}\|_{\mathbf{y}})$	1.0	30.0

Table B. The loss weights for different loss terms to balance the effect of each term used in our experiments.

them from the ground-truth frames in any case.

To obtain the keypoint heatmaps, we first linearly upscale all real and generated videos to 128×128. We feed them to OpenFace (Baltrušaitis et al., 2016) keypoint extractor, obtaining 68 keypoints for each frame. Then, we re-scale the keypoint coordinates so that they can fit into 64×64 frames (instead of 128×128). For each keypoint, We generate a Gaussian heatmap with the variance of 1/28. Then, for each frame, we merge the 68 heatmaps (64×64 pixels) into a single channel by taking the maximum value pixel-wisely. The heatmaps for the last 28 frames are concatenated channel-wisely.

We use a 2D CNN architecture of `conv2d(32) – BN – leakyReLU – conv2d(32) – BN – leakyReLU – conv2d(32) – BN – leakyReLU – conv2d(32) – BN – leakyReLU – conv(32) – BN – leakyReLU`, where `BN` is batch normalization (Ioffe & Szegedy, 2015) and `conv2d(`k`)` is a 2D convolutional layer (Tran et al., 2015) with k filters of 4×4 kernel. We use stride 2 for the first four `conv2d` layers and stride 4 for the last one. The output is an embedding $\phi(\tilde{\mathbf{x}})$ of size 32. We feed it into `fc(32) – BN – leakyReLU – fc(6) – BN – leakyReLU – dropout(0.5) – fc(6) – softmax`.

References

Aifanti, N., Papachristou, C., and Delopoulos, A. The MUG Facial Expression Database. In *WIAMIS*, 2010.

Baltrušaitis, T., Robinson, P., and Morency, L.-P. OpenFace: An Open Source Facial Behavior Analysis Toolkit. In *WACV*, 2016.

Ekman, P. An Argument for Basic Emotions. *Cognition & Emotion*, 6(3-4):169–200, 1992.

Fasel, B., Monay, F., and Gatica-Perez, D. Latent Semantic Analysis of Facial Action Codes for Automatic Facial Expression Recognition. In *MIR*, 2004.

Friesen, W. V. and Ekman, P. EMFACS-7: Emotional Facial Action Coding System. 1983.

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015.

Odena, A., Dumoulin, V., and Olah, C. Deconvolution and Checkerboard Artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and chun Woo, W. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *NIPS*, 2015.

Song, Y., Demirdjian, D., and Davis, R. Tracking Body and Hands for Gesture Recognition: Natops Aircraft Handling Signals Database. In *FG*, 2011.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. Learning Spatiotemporal Features with 3D Convolutional Networks. In *ICCV*, 2015.