CECNN: A Convergent Error Concealment Neural Network for Videos

Razib Iqbal[1], Shashi Khanal[1] and Mohammad Kazemi[2]
[1]*Computer Science Department, Missouri State University, Springfield, Missouri, U.S.A.*
[2]*Department of Electrical Engineering, University of Isfahan, Isfahan, Islamic Republic of Iran*

Keywords: Convolutional Neural Network, Error Concealment, Transfer Learning, Video, Voxel Information.

Abstract: In video error concealment, we estimate any missing information in the video frames as close to the actual data. In this paper, we present a video error concealment technique, named Convergent Error Concealment Neural Network (CECNN), based on Convolutional Neural Network (CNN). CECNN is a two-stage process where it first learns to predict voxel information from the training dataset. It then applies transfer learning using the pre-trained model from the first stage to produce intermediate outputs. CECNN consists of dedicated paths for the past and future frames to produce the intermediate outputs, which are then combined to fill the missing information in the errored frame. The quality of the outputs from CECNN is compared with other techniques, such as motion vector estimation, error concealment using neighboring motion vectors, and generative image inpainting. The evaluation results suggest that our CECNN approach would be a good candidate for error concealment in video decoders.

1 INTRODUCTION

Video error concealment techniques refer to the estimation of missing information or lost data as close as possible to actual data in the video decoder to improve the viewers' quality of experience. In conventional methods, e.g., motion vector estimation (Tsekeridou and Pitas, 2000) or using the neighboring motion vectors (Chen et al., 1997), the lost data is replaced by predicting the missing data using the previously reconstructed error-free frames or the error-free neighboring blocks of the frame. However, the current advancements in deep learning techniques, such as Convolutional Neural Network (CNN), have opened new avenues for researchers to investigate alternatives, e.g., (Xiang et al., 2019) and (Mahmud et al., 2018), to conventional error concealment techniques.

This paper presents a novel CNN-based video error concealment network called Convergent Error Concealment Neural Network (CECNN). Unlike the commonly used single path neural networks, CECNN comprises separate paths for preceding and succeeding frames. Various image and video datasets consisting of a variety of objects and backgrounds can train our CECNN model to conceal errors in video frames with the help of transfer learning (Torrey and Shavlik, 2009). Transfer learning helps to utilize knowledge gained from the previous dataset to extract unseen features in the new dataset. Therefore, CECNN does not need re-training from scratch to train the model with various datasets. This approach speeds up the learning process and offers a more accurate and efficient model for error concealment in any previously unseen video.

The rest of this paper is organized as follows: Section 2 provides a brief overview of the related works. Section 3 covers our proposed approach. In Section 4, we present our experimental setup and results. Finally, we give our concluding remarks in Section 5.

2 LITERATURE REVIEW

Traditional error concealment techniques can be divided into spatial and temporal domain techniques, e.g., (Aign and Fazel, 1995), where the lost area is concealed using spatially neighboring pixels or available information from the past or future frames are used, respectively. Switching between spatial and temporal domain techniques have also been discussed in some papers, e.g., (Ho and Chang, 2014). However, these techniques might not be effective if the lost area is large. Another approach considered is motion vector estimation during error concealment in the decoder, e.g., (Tsekeridou and Pitas, 2000) and (Shirani et al., 2000b), similar to the motion estima-

Iqbal, R., Khanal, S. and Kazemi, M.
CECNN: A Convergent Error Concealment Neural Network for Videos.
DOI: 10.5220/0011275800003289
In *Proceedings of the 19th International Conference on Signal Processing and Multimedia Applications (SIGMAP 2022)*, pages 23-31
ISBN: 978-989-758-591-3; ISSN: 2184-9471

tion in a video encoder. Error concealment based on neighboring motion vectors, e.g., (Chen et al., 1997), assumes that the surrounding motion vectors to the lost macroblock (MB) are available. Error concealment using side information, e.g., (Hadizadeh et al., 2013), sends an additional low-resolution version of the image frame as side information to assist error concealment. Pixel-wise post-processing technique, e.g., (Atzori et al., 2001), is another form of error concealment where inside the loss concealed area, MBs are refined using mesh-based warping. In error concealment with error propagation, e.g., (Usman et al., 2016), a missing frame between two received frames is interpolated using motion trajectory and then the error concealment quality is improved by adaptive filtering. Furthermore, shape preservation loss concealment techniques, e.g. (Shirani et al., 2000a), aim to recover the object's shape in the lossy frames.

In recent years, researchers have been focusing on the use of deep neural network for video error concealment. For example, the FINNiGAN model (Koren et al., 2017) uses generative adversarial network (GAN) while performing frame interpolation. Similarly, a GAN consisting of one completion network and one discriminator network is used by the authors in (Xiang et al., 2019) that follows an encoder-decoder structure. Likewise, an adversarial learning framework using conditional GAN (cGAN) is proposed in (Mahmud et al., 2018) to reconstruct a frame when one or more frames are missing in a multi-camera scenario. However, the FINNiGAN model produces some unrelated details while trying to fill in details within the high motion region of the video, and the GAN in (Xiang et al., 2019) only uses temporal information from the past frames and omits the information from future frames.

Other works involving neural networks include image inpainting. The authors in (Liu et al., 2018) proposed the use of a partial convolution layer with an automatic mask update. However, this model does not work well in images which has thin structured objects in it, e.g., handlebars on the door. The authors in (Radford et al., 2015) introduced a different class of CNN and called it DCGAN, which works well for an image classification task but not for regression tasks like video error concealment. Similarly, the authors in (Yu et al., 2018) proposed a feed-forward CNN which can process images with multiple loss at arbitrary locations and with variable sizes. It is an enhancement of baseline generative image inpainting network (Iizuka et al., 2017) which has shown promising visual results for inpainting images of faces, building facades, and natural images. However, these image inpainting techniques consider spatial information and do not use knowledge of temporal information in video sequences. The approach in (Sankisa et al., 2018) combines convolutional long short-term memory (LSTM) model and simple convolutional layers which predict optical flow using the existing optical flows of the previous frames. However, the model needs to know the location of the error in the frame and it only uses frames from the past to train the model. Similarly, the authors in (Sankisa et al., 2020) presented a deep learning framework using capsule network architecture that uses motion as an instantiation parameter to encode motion in videos followed by motion-compensated error concealment using the extracted motion. However, this network model has been demonstrated to work with video sequences from the same training dataset. Very recently, a flow-based video completion algorithm is proposed in (Gao et al., 2020) which maintains the sharpness of the video but it produces arbitrary content in large missing regions within the video frames. Similarly, the authors in (Zeng et al., 2020) proposed a joint Spatial-Temporal Transformer Network (STTN) for video inpainting to concurrently fill lost regions in all video frames. However, STTN fails to generate accurate contents to fill lost regions in the video frames which have motion contents. Finally, a video inpainting method is proposed in (Liu et al., 2021) which aligns the frames at a feature level via implicit motion estimation and aggregates temporal features to synthesize missing content by aligning reference frames with target frame. However, this method is not suitable for practical applications.

From the above discussion, we can see that there are existing methods presented by different authors where GANs, inpainting models, and architectures of CNNs are used for video error concealment. However, to the best of our knowledge, there are no works or experiments in video error concealment using CNN architecture which uses information from both the past and future video frames. Also, the existing approaches did not consider transfer learning to make use of spatial and temporal information from both the past and future frames to conceal errors in video data.

3 PROPOSED APPROACH

Figure 1 and Figure 2 jointly represent our CECNN approach. In Figure 1, we show the model training stage. In this first stage, both the original images and video frames from the training datasets along with simulated errors (such as missing blocks and slices) in those images/frames are passed to the network and voxel information of error concealed image/frame is

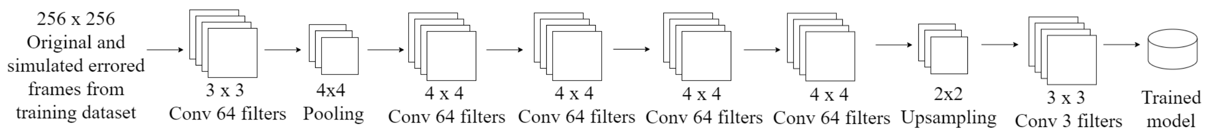

Figure 1: CECNN training stage.

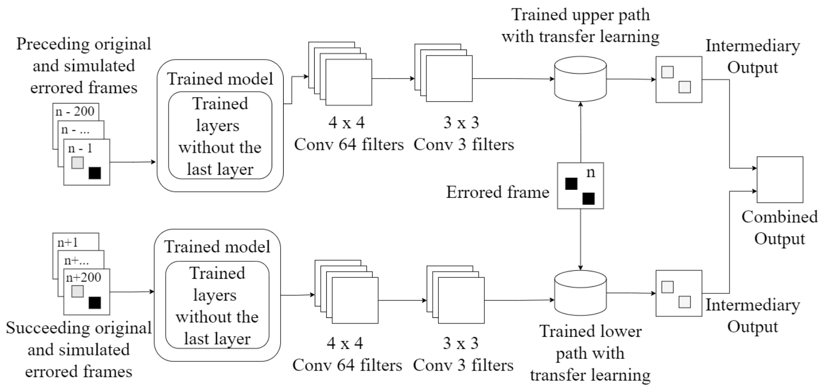
Preceding original and simulated errored frames
n - 200
n - ...
n - 1
Trained model
Trained layers without the last layer
4 x 4 Conv 64 filters
3 x 3 Conv 3 filters
Trained upper path with transfer learning
Intermediary Output
Errored frame
n
Combined Output
n+1
n+...
n+200
Succeeding original and simulated errored frames
Trained model
Trained layers without the last layer
4 x 4 Conv 64 filters
3 x 3 Conv 3 filters
Trained lower path with transfer learning
Intermediary Output

Figure 2: Trained CECNN with upper and lower path models connected via transfer learning.

predicted. During this training stage, the neural network performs regression tasks by operating in the temporal and spatial dimensions to learn voxel-related features. The CECNN training stage has eight layers, including six convolutional layers, a pooling layer, and an upsampling layer. Except for the last output layer, all the convolutional layers use relu as the activation function. The convolution is computed as per Equation 1. Equation 1 is the summation of dot products of the input image and filter values. The first layer of CECNN is a convolutional layer with 64 filters of size 3×3, where 3×3 is the height and width of convolutions, respectively. It creates feature maps that help identify different features in an image like angles, vertical or horizontal lines, edges, etc., by convolving the filter over the input image. The second layer is a max-pooling layer of size 4×4 which can be expressed as per Equation 2. In Equation 2, I_x is input, P is pooling window size, and S is the stride. This layer steadily reduces the spatial size of the feature maps produced by the previous layer, and it reduces the number of parameters and computations in our network. This reduced dimensionality of feature maps or image representation is later increased in the seventh layer. The second layer is followed by the third, fourth, fifth, and sixth layers, which are all convolutional layers and each with 64 filters of size 4×4, where 4×4 is the height and width of convolutions. We stacked four convolutional layers because it allows hierarchical decomposition of the dimensionally reduced feature maps and increases our network's performance. The authors in (Sainath et al., 2013) also attained a similar result by stacking convolutional layers. The seventh layer is an upsampling layer of size 2×2 denoted by Equation 3.

$$y[n] = x[n] * h[n] = \sum_k x[k] * h[n+k], k\varepsilon[-\infty, +\infty] \quad (1)$$

$$\left[\frac{I_x - P}{S}\right] + 1 \quad (2)$$

$$O[x', y'] = I[(int)(x+0.5), (int)(y+0.5)], x = \frac{x'}{K}, y = \frac{y'}{K} \quad (3)$$

In Equation 3, $O[x', y']$ is upsampled output, $I[x, y]$ is input, K is an upsampling factor, and (int) denotes interpolation, which is followed by the last layer. Our model reconstructs the dimensionally reduced image representation during upsampling in the seventh layer. The last layer is the output layer which is also a convolutional layer with three filters of size 3×3 and uses sigmoid as the activation function. It receives upsampled image representation from the previous layer and produces the output image of dimension equal to that of the input image, which is 256×256×3 where 256×256 is height, width, and 3 is the number of color channels, respectively. The last layer convolves the upsampled image representation with three filters of size 3×3 so that it produces the output image of dimension equal to that of the input image. The three filters produce three color components of the error concealed image: red, blue, and green. After the training, the model is saved for the error concealment stage.

In Figure 2, we present the error concealment process using transfer learning. Transfer learning (Torrey and Shavlik, 2009) is a technique in machine learning where a neural network model produced for a task is reused as the starting position for a model in another task. We use it in CECNN to re-purpose the already trained model from the training stage. We deploy two separate paths in this stage, one for the past frames and one for the future frames with respect to the errored frame, so that the outputs from each of these two paths can be used to get the final error concealed video frame. For the upper path, past frames with artificial errors and the original past frames are given as input, and the voxel information is predicted. Similarly, future frames with artificial errors and the original future frames are passed into the lower path of our network. The motivation for including a separate path for future frames in our proposed approach is to make CECNN use temporal information from the succeeding frames because the actual information missing in the errored frame can be temporally similar to its succeeding frames. During the transfer learning process, at first, the same pre-trained model from the training stage is used as starting point for both the upper and lower paths. Likewise, both upper and lower paths are completed by stacking more convolutional layers at the end of pre-trained models in each path. While forming these paths, we do not include the last (output) layer from the saved model from the previous stage because we add more convolutional layers at the end of the saved models. Furthermore, layers of the pre-trained models are frozen to reduce computational time for training. This architecture in Figure 2 can be treated as a simple non-sequentially stacked neural network model comprising of CNNs that flows through two paths and converges at a point. As can be seen in Figure 2, these saved models are both connected with two other convolutional layers - the first layer with 64 filters of size 4×4 and the other layer with three filters of size 3×3. The newly connected first layer uses relu, and the other layer uses sigmoid as the activation function. This newly formed network for transfer learning is trained again on the fly with available video frames (i.e., past and future frames with respect to the error frame being processed) with artificial errors as training data. The upper path model in Figure 2 is trained with a maximum of two hundred past frames or less if fewer past frames are available. Similarly, the lower path model is trained with a maximum of two hundred future frames or less if fewer future frames are available that come after the errored frame. A maximum of 200 preceding and 200 succeeding frames were chosen so that our model takes less training time but gets sufficient training data to reasonably produce the output. However, CECNN does not need to be trained with 200 preceding and 200 succeeding frames for each errored frame in a practical setting. Instead, a recently trained model can be used. Also, the respective parameters for the number of frames for training, frequency of training, and if the newly error concealed frames to be included in the training can be set depending on the intended use of the CECNN model.

```
Input: trained model and video frame
Output: error concealed frame
Step-1:
channel_dimension = input frame dimension;
path_model = load trained_model;
remove last layer from path_model;
freeze all the layers from path_model;
add convolutional 2D layer with 4×4 conv 64
  filter at the end of path_model;
add convolutional 2D layer with 3×3 conv
  channel_dimension filter at the end of
  path_model;
upper_path_model = path_model;
lower_path_model = path_model;
if sufficient frames available or input frame
  has error then
    load & normalize available frames as
      training data;
    simulate loss in training data;
    train upper_path_model and
      lower_path_model;
end
Step-2:
if input frame has error then
    errored_frame = load and normalize input
      frame;
    generate intermediate outputs using
      (upper_path_model, errored_frame) and
      (lower_path_model, errored_frame);
    generate and return error concealed
      output frame by combining the
      intermediate outputs;
end
```
Algorithm 1: Error concealment using transfer learning.

Now, once CECNN is trained using transfer learning, the errored video frame is passed through both upper path and lower path models, and CECNN produces two corresponding intermediate outputs. We combine the collocating data blocks from these outputs lost in the errored frame. The combination of the collocating blocks from these intermediate outputs is performed with a weighted average method (Li et al., 2017). This method uses root mean square to calculate the average of pixel values, as per Equation 4, because integer value in Red Green Blue (RGB) color code is the square root of the actual color value as per (Hoffman, 1998). Moreover, pixel-level fusion is the lowest level of image fusion that keeps more raw data as much as possible to provide rich and accurate image information, which are not provided by other fusion methods according to the authors in (Hui and Binbin, 2009). Finally, we fill the missing information in the errored frame using these combined blocks, producing the error concealed video frame. This errored frame can be in any video, from low-resolution to 4K videos. In Algorithm-1, we give the steps for error concealing using the transfer learning process shown in Figure 2.

$$R = \sqrt{\frac{R_{upper}^2 + R_{lower}^2}{2}}, G = \sqrt{\frac{G_{upper}^2 + G_{lower}^2}{2}}, B = \sqrt{\frac{B_{upper}^2 + B_{lower}^2}{2}} \quad (4)$$

In Equation 4, *upper* and *lower* represent the color of output images from upper and lower paths for red (R), green (G) and blue (B) colors.

4 PERFORMANCE EVALUATION

4.1 Dataset and Preprocessing

We used Celeba (Liu et al., 2015) and Hollywood2 (Marszalek et al., 2009) datasets. Celeba is a large-scale face attributes dataset containing more than 200K celebrity facial images of size 256×256 pixels. Hollywood2 is a dataset with 12 types of human actions (e.g., eating, fighting, running) and 10 types of scenes (e.g., house, shop, restaurant) distributed over 3669 video clips. These video clips were converted to thousands of images of size 256×256 pixels before training to match the Celeba dataset. We used OpenCV to extract and convert video frames into 256×256 sized images. However, the dimension of the input video can be of any valid dimension for training and error concealment.

4.2 Training

CECNN can be trained with any number of datasets. For the results reported in this paper, we selected the first 10,000 images from the Celeba dataset and 200 video clips from the Hollywood2 dataset. For training, at first, 5 to 15 random blocks of size ranging from 5×5 pixels to 30×30 pixels, then 5 to 15 random slices of the image of size 256 (width) × 8 (height) were randomly removed from all the frames. The number and sizes of the lost blocks were arbitrarily chosen to replicate the data loss scenario as per (Yu et al., 2018). During the training, individual frames from the video data and their corresponding errored video data are passed into the network as input. The upper- and lower-paths of the network model are trained with past and future frames, respectively. Unlike original works on CNN used explicitly for classification, we are using CNN for the regression task, which is video error concealment. Therefore, our experiments did not include a classification step. Instead, the models are primarily used for modeling voxel information to fill the missing information in the errored video frame. For this reason, we used mean-squared error (MSE) as the overall loss function. Likewise, the initial learning rate was set at 0.001 and a momentum decay of 0.9 with adam optimizer, which is also the parameters used in the similar task like (Sankisa et al., 2020). The training loss stabilized after about 100 epochs. After the training of upper- and lower-path models, these two trained models are connected using transfer learning and trained again as shown in Figure 2. Finally, the errored frame is passed through the upper and lower paths to generate the error concealed frame.

4.3 Results

The objective of our performance evaluation is two folds - first, to compare the CECNN error concealment quality with two conventional error concealment techniques, MVE (Tsekeridou and Pitas, 2000) and NMV (Chen et al., 1997), and neural network-based generative image inpainting (GII) technique (Yu et al., 2018) and secondly, to show the effectiveness of introducing two separate paths for the past and the future frames. In our performance tests, we simulated errors in the video frames by removing blocks of pixels as well as dropping slices. The quality assessment metrics we used for evaluation are PSNR, MS-SSIM, and MSE. We used Keras[1] to train our model on Google Colab. GPUs like NVIDIA K80, P100, P4, T4, and V100 are provided in Google Colab. Moreover, the standard split ratio for our training and testing data is 80:20.

4.3.1 Comparison with Other Methods

Simulating Random Blocks of Error: In Figure 3, we show three sample original frames (a,b,c) from Bus and Flower videos[2], and a sample from the Hollywood2 dataset, their respective errored frames (d,e,f) where we have manually removed some blocks of size ranging from 5×5 pixels to 30×30 pixels, and error concealed frames using MVE (g,h,i), NMV (j,k,l), GII (m,n,o), and CECNN (p,q,r) methods. We chose video samples with action variations in them. For example, in Figure 3(a), a moving bus is passing by a pole. In Figure 3(b), a person is inside a moving car, and an open flower garden with a lamp post is in the

[1]https://keras.io/api/applications/
[2]http://trace.eas.asu.edu/yuv/index.html

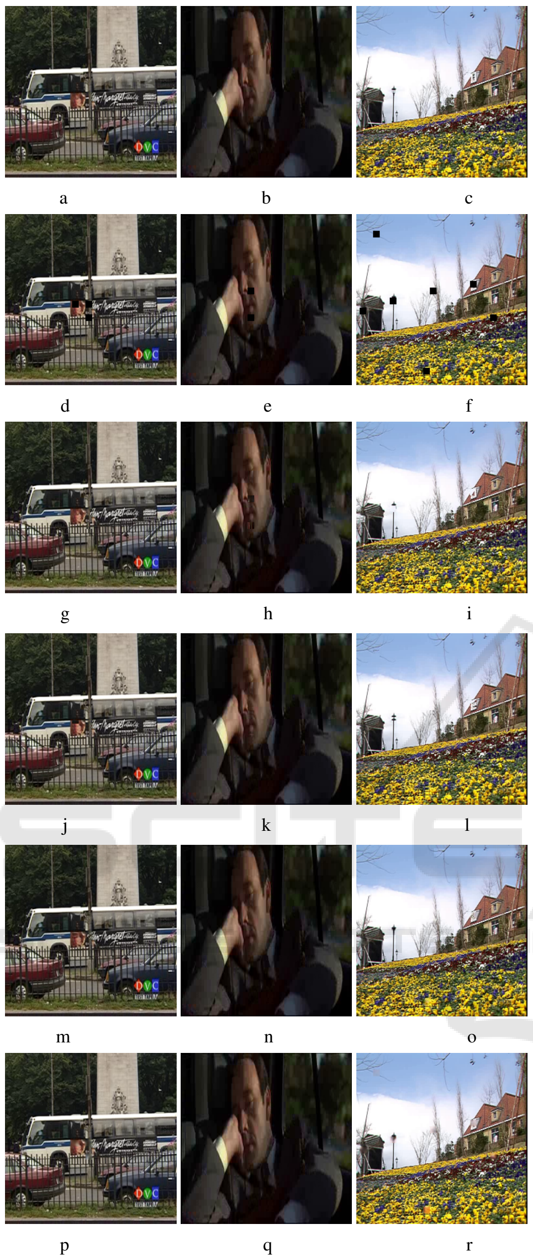

Figure 3: Original (a,b,c), errored frames (d,e,f) simulating random error blocks, error concealed frames using MVE (g,h,i), NMV (j,k,l), GII (m,n,o), and CECNN (p,q,r).

frame in Figure 3(c). We simulated random data loss at different visible areas within these original video frames. For example, there is loss of data on the pole in Figure 3(d), in between cheek, nostril, and chin on the person's face as shown in Figure 3(e), and in the lamp post, the window of the house and other locations in the flower garden as shown in Figure 3(f).

In Figure 3(g-r), the resultant error concealed frames using MVE, NMV, GII, and CECNN are presented, and their respective PSNR, MS-SSIM,

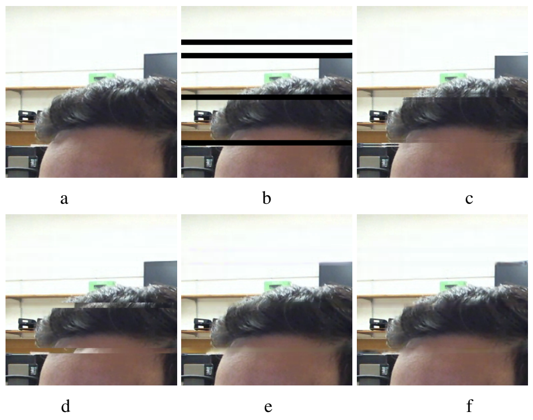

Figure 4: Original (a), errored frame (b) simulating slice loss, error concealed frame using MVE (c), NMV (d), GII (e) and CECNN (f).

and MSE values are given in Table 1. Although CECNN was trained with only Hollywood2 and Celeba datasets, we get better results with new videos (e.g. Bus and Flower) which do not belong to these datasets because we trained our network on the fly using transfer learning so that the quality of the error concealed outputs produced by CECNN remains consistent.

From Figure 3(p,q,r), we can see that our proposed CECNN produces better results than MVE and NMV, and it also produces results as good as GII while conserving the structure of the object/person in the lost part of the video frames after error concealment. For example, the structure of the woman's face and the pole is conserved in Figure 3(p) compared to Figure 3(g) and Figure 3(j). In Figure 3(q), the person's face is visually better than outputs in Figure 3(h) and Figure 3(k). Similarly, CECNN filled up the lost part of the lamp in the lamp post with lamp-like information as shown in Figure 3(r) compared to the outputs in Figure 3(i) and Figure 3(l). Finally, PSNR, MS-SSIM, and MSE values presented in Table 1 also support the above observations.

Simulating Slice Loss: In Figure 4, we show one sample original frame, its errored frame where we manually removed four slices of size 256×8 from the video frame data, and error concealed frame using MVE, NMV, GII, and CECNN. In Figure 4(a), the head of a person is moving and a portion of the head can be seen. Figure 4(b) shows simulated data loss due to lost slices in Figure 4(a). Similarly, Figure 4(c), 4(d), 4(e) and 4(f) show the error concealed frames obtained from MVE, NMV, GII, and CECNN, respectively. In Figure 4(f), we can see that error concealed output frame from our CECNN looks visually better. For quantitative analysis, we present the PSNR, MS-SSIM, and MSE values for Figure 4(c-f) in Table 2

Table 1: Error concealment quality using MVE, NMV, GII, and CECNN for random loss.

Image	Method	PSNR	MS-SSIM	MSE
Figure 3(g)	MVE	36.792244	0.998646	13.60994
Figure 3(j)	NMV	36.274521	0.997112	15.333038
Figure 3(m)	GII	38.579472	0.998062	9.018478
Figure 3(p)	CECNN	39.129711	0.998474	7.945282
Figure 3(h)	MVE	39.734142	0.996459	6.912994
Figure 3(k)	NMV	41.926872	0.997534	4.172470
Figure 3(n)	GII	34.303223	0.996068	24.141144
Figure 3(q)	CECNN	42.479977	0.998113	3.673523
Figure 3(i)	MVE	30.976776	0.991669	5.928131
Figure 3(l)	NMV	31.595686	0.992517	45.031021
Figure 3(o)	GII	47.734299	0.999495	1.095596
Figure 3(r)	CECNN	33.412643	0.994736	29.635757

Table 2: Error concealment quality using MVE, NMV, GII, and CECNN for slice loss.

Image	Method	PSNR	MS-SSIM	MSE
Figure 4(c)	MVE	28.368511	0.936281	94.673676
Figure 4(d)	NMV	28.056870	0.973456	101.716980
Figure 4(e)	GII	32.677994	0.983414	25.887426
Figure 4(f)	CECNN	35.640781	0.987977	13.086094

Table 3: Error concealment quality for upper, lower, and combined paths of CECNN.

Image	**Method**	**PSNR**	**MS-SSIM**	**MSE**
Figure 5(e)	Upper	28.4618	0.9676	68.344
Figure 5(g)	Lower	28.5565	0.9680	66.870
Figure 5(i)	Combined	28.5629	0.9679	66.772
Figure 5(f)	Upper	33.3130	0.9944	30.323
Figure 5(h)	Lower	32.8422	0.9934	33.795
Figure 5(j)	Combined	33.4126	0.9947	29.635

which shows that the CECNN gives a better result.

4.3.2 Effectiveness of using Two Paths in CECNN

During the transfer learning process (in Figure 2), as the first step, upper and lower paths are trained with preceding and succeeding video frames, respectively. Then an errored video frame is passed through both paths to obtain two different intermediate outputs - one from the upper path and another from the lower path. The final error concealed output is obtained from these two intermediate outputs by combining the collocating blocks of data from these outputs. To show the efficacy of this approach, we present two original frames from a custom video and Flower video in Figure 5(a) and Figure 5(b). In Figure 5(c) and Figure 5(d), we show errored versions of the original frames representing slice loss and random error blocks. Figure 5(e) and Figure 5(f) show the error concealed form of the errored frames in Figure 5(c) and Figure 5(d) respectively using the upper path only. Similarly, Figure 5(g) and Figure 5(h) show the error concealed form of the errored frames in 5(c) and Figure 5(d) respectively using the lower path only. Finally, error concealed output combining both the intermediate outputs from the upper and the lower paths for each errored frame are produced and shown in Figure 5(i) and Figure 5(j).

In Table 3, we give the PSNR, MS-SSIM, and MSE values for the error concealed outputs presented in Figure 5(e-j). We can see that error concealed frames combining the intermediate outputs from the upper and the lower paths give better PSNR, MS-SSIM, and MSE values compared to the upper or the lower path separately.

5 CONCLUSION

In this paper, we proposed a CNN-based video error concealment technique named CECNN. CECNN uses both the spatial information of the errored frame and

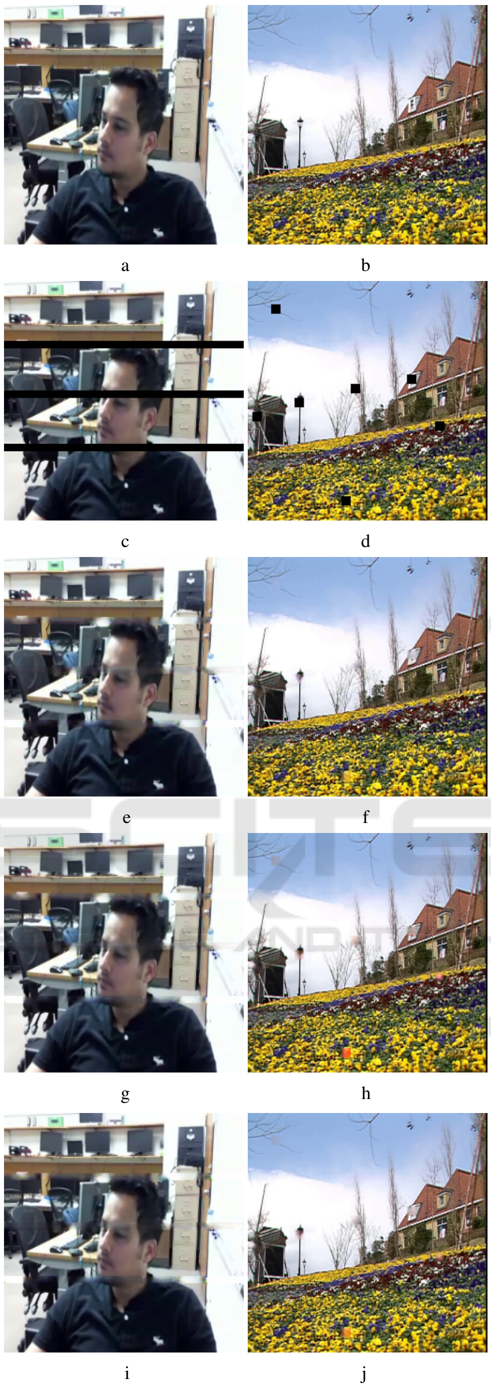

Figure 5: Original (a,b), errored frames (c,d), and error concealed frames from CECNN upper path (e,f), lower path (g,h), and combined (i,j).

the temporal information from past and future frames, unlike the existing works that only use either spatial domain information or past frames' data to predict the missing part of the errored frame. CECNN model also consists of two stages - training and transfer learning. CECNN is first trained with various image and video datasets, and then the missing information is estimated as close as possible in the errored video frames using transfer learning. This approach speeds up the learning process and offers a more accurate and efficient model for video error concealment. When trained with different datasets from various domains, the CECNN learns more variations to conceal errors more accurately as complex functional relationships between the input and output data can be learned by neural networks. Given the relatively lightweight nature of our proposed CECNN model, it would be a good candidate for error concealment in video decoders.

REFERENCES

Aign, S. and Fazel, K. (1995). Temporal and spatial error concealment techniques for hierarchical mpeg-2 video codec. *IEEE International Conference on Communications (ICC)*, 3:1778–1783.

Atzori, L., De Natale, F. G., and Perra, C. (2001). A spatio-temporal concealment technique using boundary matching algorithm and mesh-based warping (bma-mbw). *IEEE Transactions on Multimedia*, 3(3):326–338.

Chen, M.-J., Chen, L.-G., and Weng, R.-M. (1997). Error concealment of lost motion vectors with overlapped motion compensation. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(3):560–563.

Gao, C., Saraf, A., Huang, J.-B., and Kopf, J. (2020). Flow-edge guided video completion. *European Conference on Computer Vision*, pages 713–729.

Hadizadeh, H., Bajić, I. V., and Cheung, G. (2013). Video error concealment using a computation-efficient low saliency prior. *IEEE Transactions on Multimedia*, 15(8):2099–2113.

Ho, C.-L. and Chang, L.-W. (2014). Temporal and spatial error concealment using cooperative game. *International Conference on Information Science, Electronics and Electrical Engineering*, 1:380–384.

Hoffman, M. T. (1998). Image file storage and retrieval system. US Patent 5,761,655.

Hui, T. and Binbin, W. (2009). Discussion and analyze on image fusion technology. *International Conference on Machine Vision*, pages 246–250.

Iizuka, S., Simo-Serra, E., and Ishikawa, H. (2017). Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14.

Koren, M., Menda, K., and Sharma, A. (2017). Frame interpolation using generative adversarial networks.

Li, S., Kang, X., Fang, L., Hu, J., and Yin, H. (2017). Pixel-level image fusion: A survey of the state of the art. *Information Fusion*, 33:100–112.

Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., and Catanzaro, B. (2018). Image inpainting for irregular holes using partial convolutions. *European Conference on Computer Vision*, pages 85–100.

Liu, R., Weng, Z., Zhu, Y., and Li, B. (2021). Temporal adaptive alignment network for deep video inpainting. *International Joint Conferences on Artificial Intelligence*, pages 927–933.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. *IEEE International Conference on Computer Vision*, pages 3730–3738.

Mahmud, T., Billah, M., and Roy-Chowdhury, A. K. (2018). Multi-view frame reconstruction with conditional gan. *IEEE Global Conference on Signal and Information Processing*, pages 1164–1168.

Marszalek, M., Laptev, I., and Schmid, C. (2009). Actions in context. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2929–2936.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Sainath, T. N., Kingsbury, B., Mohamed, A.-r., Dahl, G. E., Saon, G., Soltau, H., Beran, T., Aravkin, A. Y., and Ramabhadran, B. (2013). Improvements to deep convolutional neural networks for lvcsr. *Workshop on automatic speech recognition and understanding*, pages 315–320.

Sankisa, A., Punjabi, A., and Katsaggelos, A. K. (2018). Video error concealment using deep neural networks. *IEEE International Conference on Image Processing (ICIP)*, pages 380–384.

Sankisa, A., Punjabi, A., and Katsaggelos, A. K. (2020). Temporal capsule networks for video motion estimation and error concealment. *Signal, Image and Video Processing*, 14(7):1369–1377.

Shirani, S., Erol, B., and Kossentini, F. (2000a). A concealment method for shape information in mpeg-4 coded video sequences. *IEEE Transactions on Multimedia*, 2(3):185–190.

Shirani, S., Kossentini, F., and Ward, R. (2000b). A concealment method for video communications in an error-prone environment. *IEEE Journal on Selected Areas in Communications*, 18(6):1122–1128.

Torrey, L. and Shavlik, J. (2009). Transfer learning. handbook of research on machine learning applications. *IGI Global*, 3:17–35.

Tsekeridou, S. and Pitas, I. (2000). Mpeg-2 error concealment based on block-matching principles. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(4):646–658.

Usman, M., He, X., Lam, K.-M., Xu, M., Bokhari, S. M. M., and Chen, J. (2016). Frame interpolation for cloud-based mobile video streaming. *IEEE Transactions on Multimedia*, 18(5):831–839.

Xiang, C., Xu, J., Yan, C., Peng, Q., and Wu, X. (2019). Generative adversarial networks based error concealment for low resolution video. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1827–1831.

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. S. (2018). Generative image inpainting with contextual attention. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514.

Zeng, Y., Fu, J., and Chao, H. (2020). Learning joint spatial-temporal transformations for video inpainting. *European Conference on Computer Vision*, pages 528–543.