TRIPLET LOSS FEATURE AGGREGATION FOR SCALABLE HASH

Wei Jia[1], *Li Li*[1], Member, IEEE, *Zhu Li*[*1], Senior Member, IEEE, *Shuai Zhao*[2], *Shan Liu*[2],

[1]University of Missouri Kansas City, [2]Tencent America

ABSTRACT

The increasing demands of high resolution and high quality videos aggravate the burden of limited cluster storage and restricted bandwidth resources. Hence, video de-duplication in storage and transmission is becoming an important feature for video cloud storage and Content Delivery Network (CDN) service providers. However, the current video de-duplication schemes mostly rely on the URL-based solution, which is unable to deal with non-cacheable content like video. The same video content with various resolutions and qualities may have completely different URL identification. In this paper, we propose a novel content-based video segmentation identification scheme that is invariant to the underlying codec and operational bit rates. It computes robust features from a triplet loss deep learning network that captures the invariance of the same content under different coding tools and strategy. In addition, a scalable hashing solution is developed based on Fisher Vector aggregation of the convolutional features from the Triplet loss network. Furthermore, we apply binary tree to obtain the triplets to improve the performance of the triplet-loss based VGG network. Our simulation results show significant improvements in terms of large scale video repository de-duplication compared with the state-of-the-art method.

Index Terms— Binary Hash, Binary Tree, Fisher Vector, Triplet Loss, Video De-duplication

1. INTRODUCTION

Modern dynamic adaptive video streaming methods such as MPEG-DASH [1], Apple HLS [2], and Microsoft Smooth Streaming [3] have significant influences on how content providers store and serve the media contents in the cloud such as Content Delivery Network (CDN). Over the Top (OTT) content providers are also pushing subscription-based video on demand (VoD) services that offer streaming services on television. The media content creation, sharing, and consumption processes generate many duplicates but are not necessarily identical in bit stream.

Video de-duplication is a necessary and promising way to alleviate the problem. However, it mainly has the following difficulties to develop a good video de-duplication algorithm. First, multimedia data from the users on the cloud cluster is valuable. Therefore, it is extremely strict to remove any videos. This results in that we should develop the system with high accuracy and recall. Especially, we focus on the true positive rate (TPR) under false positive rate (FPR) equals to 0 because we do not allow the situation of removing any non-repetitive videos. Second, a tremendous quantity of videos may cost the system much time to recognize and match the video identity if the algorithm is inefficient. A high-delay method cannot satisfy the real time requirement in the social media applications.

To alleviate the stated problems above, [4][5] basically generate a hash for each data block without considering the characteristics of the video. Feng *et al.* developed a novel hash scheme which is scalable and robust to typical CDN induced transcoding and manipulations [6]. Xu *et al.* [7] present a cache design namely DeepCache with deep learning inference. It exploits temporal similarities in videos to reuse the cached information. In addition, these works [8–13] roughly present a type of secure system architecture design which bridge together the advantages of video compression and encrypted data de-duplication. However, the hashing generated by those methods are not distinct enough to achieve high TPR under FPR equals to 0.

Therefore, we propose a novel deep learning based scheme to de-duplicate the replicated videos in the cluster in this paper. Our contribution towards video de-duplications are summarized as below:

1) We consider combining triplet loss with Visual Geometry Group (VGG) deep learning network [15] which is trained of outstanding performance by a huge dataset to derive the features. Triplet loss based network can learn convolutional features which is invariant to coding method and bit rates.

2) We propose applying fish vector [16] for feature aggregation. The fish vectors are generated from the outputs of VGG with triplet loss.

3) We propose employing binary tree to obtain the triplets to boost the performance of the triplet-loss based VGG network.

4) We utilize the extracting algorithm to generate a scalable binary hash. The scalable binary hash can obtain different trade-offs to satisfy different bitrate requirements.

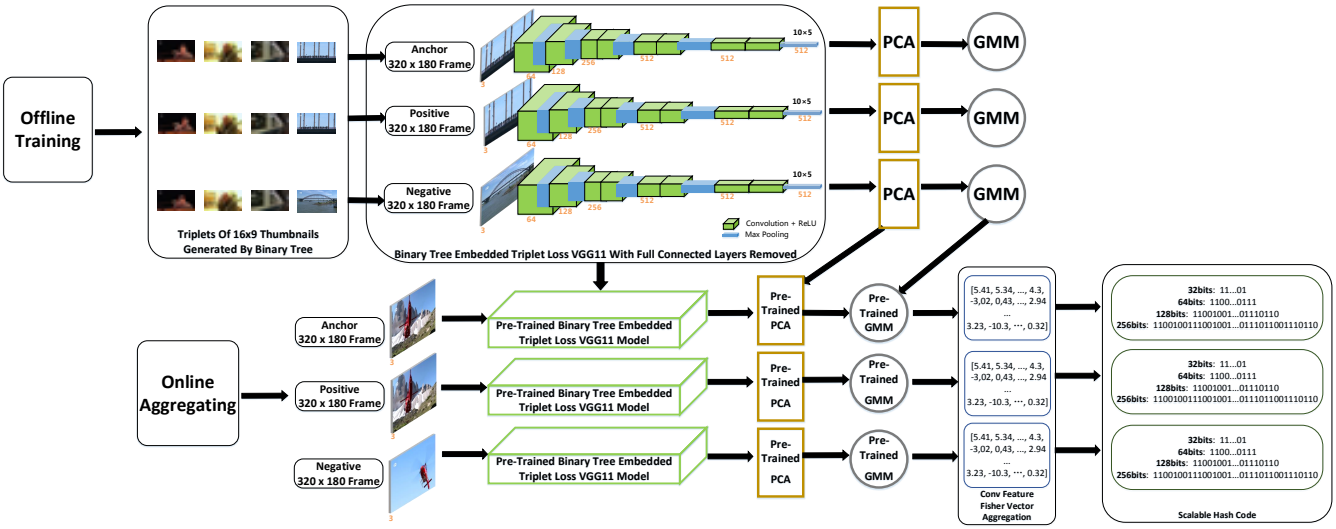

Fig. 1. Scalable hashing framework. Offline training: We first develop the binary-tree generating valuable triplets including anchors, positive samples and negative samples; Then we input the 320×180 frames of triplets into VGG11 of full connected layers removed respectively to train the network, PCA and GMM [14] models. Online aggregating: we utilize the trained network, PCA and GMM models to calculate the fisher vector aggregation from conv features; then we quantize the fisher vectors to scalable binary hash code involving $32, 64, 128$ and 256 bits.

The experimental results show that the proposed binary-tree embedded triplet loss network combing with scalable hash from fisher vector (BTF) approach outperforms the cross-entropy [17] function with PCA [18] (CP) approach in various scalable hashes.

The remaining of this paper is organized as follows. We show the triplet loss based scalable hash framework in Section 2. In Section 3, the Binary-Tree algorithm to produce the triplets with similar variance attributions is introduced in detail. The experimental results are shown in Section 4. We conclude the paper in Section 5.

2. TRIPLET LOSS BASED SCALABLE HASHING FRAMEWORK

The overall framework of the proposed scalable hash scheme is illustrated in Fig. 1. It consists of two components: a) Triplet loss network feature representation generation in subsection 2.1; b) The fish vector (FV) feature aggregation using fish vector for generating scalable hash in subsection 2.2.

2.1. Triplet Loss Network

The main difference of the proposed network compared with the original VGG is to use the triplet loss [19] to replace the cross-entropy loss to make the feature more distinguishable and more suitable for video de-duplication. In order to obtain a reliable triplet loss embedded network, three basic constraints shall be applied to choose the triplets: First, an anchor feature x_j^a should be as close to the same type samples x_j^p as possible. Second, an anchor feature x_j^a should be as far away from other types samples x_j^n as possible. Third, the distance between anchor feature x_j^a with positive feature x_i^p should be less than the one between anchor feature x_i^a and negative feature x_j^n with a marginal distance.

Therefore, we employ the accuracy indicator in (1) as our loss function,

$$\alpha_{\iota_{ap},\iota_{an}}(i) = \frac{\sum_i^N \Phi(\iota_{ap}(i) - \iota_{an}(i) - \xi)}{\Psi_{a,p,n}}, \tag{1}$$

where ι_{ap} is the L_2 distance between positive pairs which mean anchors and positive samples pairs. ι_{an} is the L_2 distance between negative pairs which mean anchors and negative samples pairs. ξ is the least margin distance between ι_{ap} and ι_{an} which makes sure the base judgment is correct. Here we set ξ equal 1.0. We train our triplet loss based network using the dataset that will be described in Section 3 for 81 epochs. We choose the best validation checkpoint in the $31st$ epoch with accuracy of 92.08% to derive the features.

2.2. FV Aggregation For Scalable Hash

The Fish vector is used to aggregate the convolutional features to generate the scalable hash. Note that the fisher vector selection is vital for scalable hash. We first derive the eigenvalues from trained PCA models and covariance weights from trained GMM models using the features generated by the triplet loss based network. We then normalize the eigenvalues of PCA, and use a weighted combination to generate

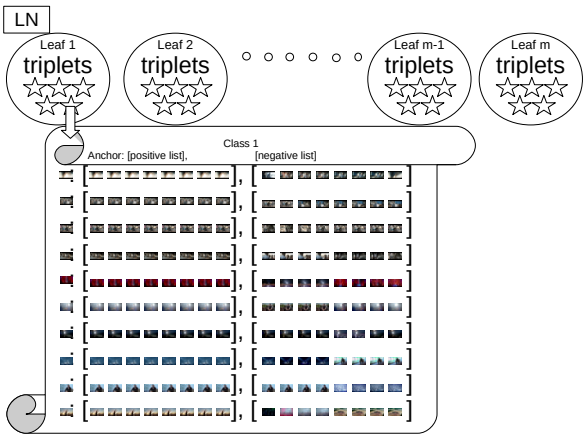

Fig. 2. Binary-Tree generates hard valuable triplets. Each thumbnail represents its coded frame. We employ the Binary-Tree diving the thumbnail samples according to their attributions of components. Thus we select the thumbnails of similar features as the hard triplet comprising positive samples and negative samples for anchor.

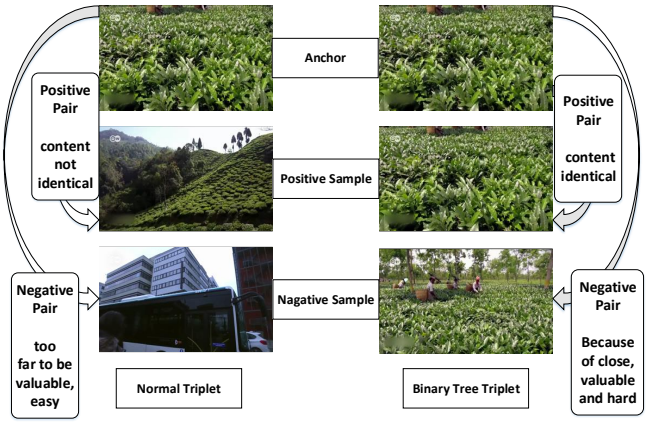

Fig. 3. Comparison between Normal triplet and Binary-Tree triplet. The binary tree triplet exhibits similar features on its positive sample and negative sample. Especially for the negative sample, due to the close distance with anchor, it provides the network with the difficult case and improve its robustness. But normal triplet shows far distance with anchor, it is easy to study by VGG11.

the scalable hash,

$$\Xi(m,n) = \alpha \times \varepsilon(kd) + (1-\alpha) \times W(nc), \qquad (2)$$

where $\varepsilon(m)$ is the normalized eigenvalues matrix [20] PCA with kd components. W(n) represents the normalized covariance weights matrix of GMM with nc components. α means the weighted coefficient ranging from 0 to 1. We choose the first τ value of Ξ as the main features constituting scalable hash code. τ is the length of hash code, such as $32, 64, 128$ and 256 bits.

3. TRIPLETS GENERATION

In this section, we will introduce how binary tree divides the dataset and generates the triplets.

3.1. Train Sequence Selection

To maximize the robust and learning capability of the deep learning network and models, we first generate a dataset with a large number of videos. First, we collect 177.7 hours original documentary [21] videos that have varieties of scenarios. They are all $1080p$ resolutions with $QP23$. To generate videos with different resolutions and qualities, we convert the original video to other 6 versions including $720p$ $QP23$, $720p$ $QP28$, $720p$ $QP33$, $480p$ $QP23$, $480p$ $QP28$, $480p$ $QP33$ using ffmpeg. In this way, for the same video content, we can have 7 versions in total. We then sample the 7 versions videos to 320×180 frames and 16×9 thumbnails. The 320×180 frames will be used for training the triplet network, while the thumbnails will be used for training the GMM model. We extract the thumbnails 2 frames per second such that 1-hour video converts to 7,200 frames (or thumbnails).

3.2. Binary Tree Based Triplets Generation

In this work, as drawn in Fig. 2, we choose Binary-Tree [22] to split the huge video frames dataset to analyze out valuable and hard triplets. We use thumbnails proposed in Section 3.1 as input data for producing triplets with Binary-Tree. The same thumbnails belonging to the same leaf node are used to derive the positive pairs. Different thumbnails belonging to the same leaf node are used to derive the negative pairs. Since the thumbnails in the same leaf node are similar to each other, we can generate more distinctive triplets in this way. Fig. 3 compares the normal triplet with the Binary-Tree generated triplet. We can see that the triplets generated by the binary-tree are more valuable and distinct to learn.

4. EXPERIMENTAL RESULTS

To demonstrate the effectiveness of the proposed binary-tree embedded triplet loss network combing with scalable hash from fisher vector (BTF) approach, we compare it with the cross-entropy loss network combined with PCA (CP). To be more specific, the VGG11 model pre-trained on ImageNet using cross-entropy loss combined with PCA is used as the anchor. The VGG11 model trained on our training set using

Table 1. Scalable hash code TPR comparison between CP and our BTF when FPR = 0

CP TPR— our BTF TPR	32 bits	64 bits	128 bits	256 bits
kd 16 - - — kd 16, nc 24, α 0.01	0.3216|**0.7454**	0.5171|**0.8835**	0.7140|**0.9190**	0.8073|**0.9445**
kd 16 - - — kd 16, nc 24, α 0.2	0.3216|**0.7290**	0.5171|**0.8607**	0.7140|**0.9069**	0.8073|**0.9461**
kd 16 - - — kd 16, nc 24, α 0.4	0.3216|**0.5407**	0.5171|**0.7802**	0.7140|**0.9095**	0.8073|**0.9399**
kd 16 - - — kd 16, nc 24, α 0.8	**0.3216**|0.2473	0.5171|**0.6904**	0.7140|**0.8785**	0.8073|**0.9302**
kd 16 - - — kd 16, nc 24, α 0.99	**0.3216**|0.2614	0.5171|**0.6797**	0.7140|**0.8630**	0.8073|**0.9188**
kd 16 - - — kd 16, nc 48, α 0.01	0.3216|**0.6928**	0.5171|**0.8342**	0.7140|**0.9188**	0.8073|**0.9409**
kd 16 - - — kd 16, nc 48, α 0.2	0.3216|**0.7590**	0.5171|**0.8685**	0.7140|**0.9180**	0.8073|**0.9421**
kd 16 - - — kd 16, nc 48, α 0.4	0.3216|**0.7316**	0.5171|**0.8457**	0.7140|**0.9090**	0.8073|**0.9338**
kd 16 - - — kd 16, nc 48, α 0.8	0.3216|**0.7371**	0.5171|**0.8516**	0.7140|**0.8904**	0.8073|**0.9295**
kd 16 - - — kd 16, nc 48, α 0.99	0.3216|**0.7369**	0.5171|**0.8519**	0.7140|**0.8878**	0.8073|**0.9278**
kd 24 - - — kd 24, nc 24, α 0.01	0.1319|**0.6440**	0.3359|**0.7007**	0.5569|**0.8392**	0.7323|**0.9459**
kd 24 - - — kd 24, nc 24, α 0.2	0.1319|**0.5809**	0.3359|**0.7497**	0.5569|**0.8697**	0.7323|**0.9557**
kd 24 - - — kd 24, nc 24, α 0.4	0.1319|**0.4935**	0.3359|**0.7559**	0.5569|**0.8721**	0.7323|**0.9590**
kd 24 - - — kd 24, nc 24, α 0.8	0.1319|**0.4916**	0.3359|**0.7288**	0.5569|**0.8859**	0.7323|**0.9595**
kd 24 - - — kd 24, nc 24, α 0.99	0.1319|**0.4883**	0.3359|**0.7347**	0.5569|**0.8478**	0.7323|**0.9569**
kd 24 - - — kd 24, nc 48, α 0.01	0.1319|**0.6592**	0.3359|**0.7835**	0.5569|**0.9092**	0.7323|**0.94**
kd 24 - - — kd 24, nc 48, α 0.2	0.1319|**0.5526**	0.3359|**0.7673**	0.5569|**0.8845**	0.7323|**0.9516**
kd 24 - - — kd 24, nc 48, α 0.4	0.1319|**0.2169**	0.3359|**0.5669**	0.5569|**0.8447**	0.7323|**0.9442**
kd 24 - - — kd 24, nc 48, α 0.8	**0.1319**|0.1049	0.3359|**0.5921**	0.5569|**0.8259**	0.7323|**0.9333**
kd 24 - - — kd 24, nc 48, α 0.99	**0.1319**|0.0966	0.3359|**0.5923**	0.5569|**0.8202**	0.7323|**0.9333**

triplet less combined with scalable hash from fisher vector is used as the test. We test two cases with $kd = 16$ and $kd = 24$ for both the anchor and the proposed algorithm. We also test two cases with $nc = 24$ and $nc = 48$ for GMM. The α is set as 0.01, 0.2, 0.4, 0.8 and 0.99 in this experiment. We test $32, 64, 128$ and 256 as the numbers of scalable hash bits.

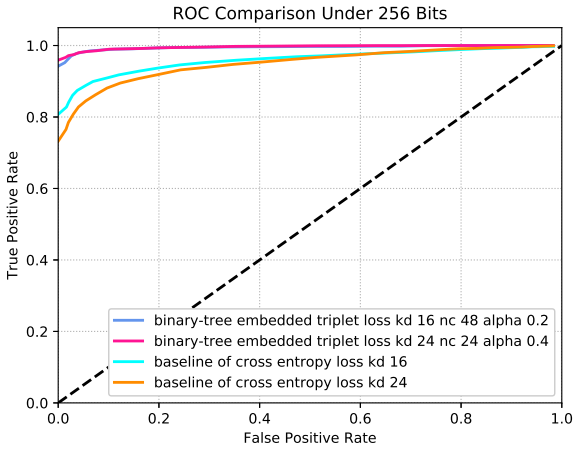

Fig. 4. Comparison results of scalable hash ROC between CP method and proposed BTF approach under 256 bits. Our BTF scheme shows an overwhelming advantage of ROC compared to CP method on each scalable hash bits. Typically, the ROC curves of BTF hold a high level momentum of TPR that is above 0.9421.

Table 1 shows the comparison between the proposed BTF approach and the CP method. We can see that the proposed BTF algorithm outperforms the CP method significantly in all the test cases. When kd equals 16, our BTF exhibits the best performance of 0.9461 TPR under nc 24, α 0.2 and 256 bits. It is 0.1388 higher than the best result 0.8073 of CP. When kd equals 24, our BTF shows the best performance of 0.9595 TPR under nc 24, α 0.8 and 256 bits. It is 0.2272 higher than the best result 0.7323 of CP. The experimental results obviously demonstrate the effectiveness of the proposed BTF.

We also compare the integrated TPR trend as FPR increases in the receiver operating characteristic (ROC) [23], as described in Fig. 4. Due to the limited space, we select nc 48 α 0.2 and nc 24 α 0.4 as representations of kd 16 and 24 respectively for the proposed BTF approach. The ROC curves also show that the proposed BTF approach outperforms the CP method significantly.

5. CONCLUSION

In this paper, we propose a distinct video de-duplication framework involving a triplet loss network learning convolutional features. In addition, we generate the scalable hash from fish vector aggregation of the convolutional features. Furthermore, we design a novel binary tree embedded algorithm to generate hard triplet samples for triplet loss function feeding VGG network more robustly. The experimental results demonstrate the effectiveness of the proposed BTF algorithm.

6. REFERENCES

[1] Iraj Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[2] M Christopher, "Mpeg-dash vs. apple hls vs. microsoft smooth streaming vs. adobe hds," 2015.

[3] Alex Zambelli, "Iis smooth streaming technical overview," *Microsoft Corporation*, vol. 3, pp. 40, 2009.

[4] John Edward Gerard Matze, "System and method for data deduplication," June 19 2012, US Patent 8,205,065.

[5] Emmanuel Barajas Gonzalez, Shaun E Harrington, David C Reed, and Max D Smith, "Efficient video data deduplication," May 9 2017, US Patent 9,646,017.

[6] Shan Feng, Zhu Li, Yiling Xu, and Jun Sun, "Compact scalable hash from deep learning features aggregation for content de-duplication," in *Multimedia Signal Processing (MMSP), 2017 IEEE 19th International Workshop on*. IEEE, 2017, pp. 1–5.

[7] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu, "Deepcache: Principled cache for mobile deep vision," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 2018, pp. 129–144.

[8] Yifeng Zheng, Xingliang Yuan, Xinyu Wang, Jinghua Jiang, Cong Wang, and Xiaolin Gui, "Enabling encrypted cloud media center with secure deduplication," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. ACM, 2015, pp. 63–72.

[9] Yifeng Zheng, Xingliang Yuan, Xinyu Wang, Jinghua Jiang, Cong Wang, and Xiaolin Gui, "Toward encrypted cloud media center with secure deduplication," *IEEE Transactions on Multimedia*, vol. 19, no. 2, pp. 251–265, 2017.

[10] Fatema Rashid, Ali Miri, and Isaac Woungang, "Proof of storage for video deduplication in the cloud," in *2015 IEEE International Congress on Big Data*. IEEE, 2015, pp. 499–505.

[11] Fatema Rashid, Ali Miri, and Isaac Woungang, "A secure video deduplication scheme in cloud storage environments using h. 264 compression," in *2015 IEEE First International Conference on Big Data Computing Service and Applications*. IEEE, 2015, pp. 138–146.

[12] Hongyang Yan, Xuan Li, Yu Wang, and Chunfu Jia, "Centralized duplicate removal video storage system with privacy preservation in iot," *Sensors*, vol. 18, no. 6, pp. 1814, 2018.

[13] Xuan Li, Jie Lin, Jin Li, and Biao Jin, "A video deduplication scheme with privacy preservation in iot," in *International Symposium on Computational Intelligence and Intelligent Systems*. Springer, 2015, pp. 409–417.

[14] Douglas Reynolds, "Gaussian mixture models," *Encyclopedia of biometrics*, pp. 827–832, 2015.

[15] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] Philippe-Henri Gosselin, Naila Murray, Hervé Jégou, and Florent Perronnin, "Revisiting the fisher vector for fine-grained classification," *Pattern recognition letters*, vol. 49, pp. 92–98, 2014.

[17] Reuven Y Rubinstein and Dirk P Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*, Springer Science & Business Media, 2013.

[18] Ian Jolliffe, "Principal component analysis," in *International encyclopedia of statistical science*, pp. 1094–1096. Springer, 2011.

[19] De Cheng, Yihong Gong, Sanping Zhou, Jinjun Wang, and Nanning Zheng, "Person re-identification by multi-channel parts-based cnn with improved triplet loss function," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1335–1344.

[20] J Li and L Ji, "Adjusting multiple testing in multilocus analyses using the eigenvalues of a correlation matrix," *Heredity*, vol. 95, no. 3, pp. 221, 2005.

[21] DW Documentary, "Dw documentary," `https://www.youtube.com/channel/UCW39zufHfsuGgpLviKh297Q`, Accessed Oct , 2018.

[22] Michael Greenspan and Mike Yurick, "Approximate kd tree search for efficient icp," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.* IEEE, 2003, pp. 442–448.

[23] Caren M Rotello, Evan Heit, and Chad Dubé, "When more data steer us wrong: Replications with the wrong dependent measure perpetuate erroneous conclusions," *Psychonomic Bulletin & Review*, vol. 22, no. 4, pp. 944–954, 2015.