Nuisance-Label Supervision: Robustness Improvement by Free Labels

Xinyue Wei[1], Weichao Qiu[2], Yi Zhang[2], Zihao Xiao[2], and Alan Yuille[2]

[1]University of California San Diego
[2]Johns Hopkins University

Abstract

In this paper, we present a Nuisance-label Supervision (NLS) module, which can make models more robust to nuisance factor variations. Nuisance factors are those irrelevant to a task, and an ideal model should be invariant to them. For example, an activity recognition model should perform consistently regardless of the change of clothes and background. But our experiments show existing models are far from this capability. So we explicitly supervise a model with nuisance labels to make extracted features less dependent on nuisance factors. Although the values of nuisance factors are rarely annotated, we demonstrate that besides existing annotations, nuisance labels can be acquired freely from data augmentation and synthetic data. Experiments show consistent improvement in robustness towards image corruption and appearance change in action recognition.

1. Introduction

Model robustness is an important topic because it decides on whether a model can work well in actual applications. Many factors can affect model robustness, and **spurious correlation** is one of them. Spurious correlation refers to using inaccurate information to make predictions. For example, *playing football* and *grass* are always bundled up in action datasets; if a model uses grass to predict the task label *playing football*, it is relying on the spurious correlation between *grass* and the task label, and when people play football in other scenes, it fails. For a certain task, we call factors related to the task as **essential factors** and others as **nuisance factors**. When training a model, the goal is to accurately make predictions depending on essential factors, while invariant to the change of nuisance factors. Therefore, we should prevent models from relying on spurious correlation between task labels and nuisance factors.

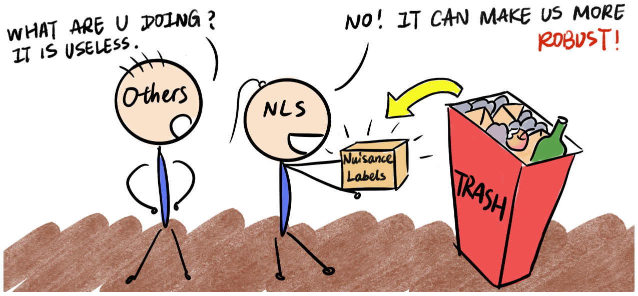

Figure 1. Simple illustration of the core concept. Our proposed method Nuisance-label Supervision (NLS) module makes use of *nuisance labels*, which is usually threw out by other methods, to improve model robustness.

Increasing data diversity is a common way to solve spurious correlation however it suffers from problems. The strong correlations are hard to break, e.g. the football and grass pair commonly appear in most action datasets. Simply adding to training data diversity is not sufficient for removing the correlation between nuisance factors and task labels. Recent works illustrate this problem that ImageNet-trained models are still biased towards textures [11], action recognition models trained on UCF101 or Kinetics are sensitive to viewpoint and human appearance changes [21]. Therefore, exploration about proactively disentangling nuisance factors and task label is worth thinking about.

We propose to explicitly use nuisance labels to train a feature representation robust to nuisance factor perturbations. We design a Nuisance-label Supervision (NLS) module, which utilizes nuisance labels as extra supervision signals and makes the whole network training in an adversary way. We first prove the core idea on MNIST-C which is widely used to study image corruption, then we demonstrate the method effectiveness on UCF101, and NTU RGB-D, which are both action datasets; the former is a popular dataset, but models trained on it usually fail to generalize to other datasets; the latter is a controlled dataset with multiple available nuisance labels. By using NLS, our trained model will be insensitive to viewpoint or human appearance changes.

Though NLS is simple and intuitive, collecting enough training data with nuisance labels is a non-trivial issue. We answer this challenging question by giving out three practical and low-cost ways of acquiring nuisance labels: 1) image processing parameters when doing data augmentation; 2) metadata in the data collection procedure; 3) rendering parameters from synthetic data generation. These three types of data are free and often ignored; Fig. 1 vividly illustrates the idea of making waste profitable. We describe in detail how we get nuisance labels from these three sources in Sec. 3.3 and demonstrate the effectiveness in the experiment section. Previously, extra labels other than the task label have been proved effective in intermediate supervision [16] and multi-task learning [38]. Compared with them, nuisance labels in our method are easy to collect because they all come from the processing steps of data generation.

Our module is generic and easy to use. It is not restricted to tasks or models and can be attached to any existing models, which makes our method widely applicable. The improvement by NLS is totally for free because the nuisance labels come from the data generation process, which costs no time to annotate. Besides, NLS only works during the training time and adds no extra calculation to inference.

In addition to NLS, we explore using nuisance labels for understanding model robustness. We diagnose model dependency on a certain nuisance factor and analyze its relationship with robustness. The results show the more a model relies on nuisance factors, the worse it performs on test set with different nuisance factor values.

The contribution can be summarized as follows:

- We propose a Nuisance-label Supervision (NLS) module to improve model robustness by breaking spurious correlation. The module is generic and can be applied to different tasks and models.
- We propose three practical and low-cost ways of acquiring nuisance labels. Besides existing annotations, parameters from data augmentation and synthetic data can also be used for generating nuisance labels.
- Extensive experiments demonstrate the effectiveness of improving model robustness to image corruption and appearance changes. Nuisance labels are used to further understand over-fitting and model robustness improvement.

2. Related Work

Robustness against irrelevant perturbations. Recent works attempt to solve generalization failure by data augmentation and bias mitigation. Cubuk and Zoph [7] proposed AutoAugment for searching improved data augmentation policies; Hendrycks and Mu [13] proposed image processing technique AugMix, Rusak and Schott [29] introduced adding Gaussian noise to training data (GNT) and novel adversarial noise generator (ANT) to overcome unseen corruptions. These approaches focus on increasing data diversity while not making use of intermediate labels. Singh *et al.* [31] used CAM [41] and feature-slitting methods to decorrelate category and its co-occurring context but only applicable to fixed category pairs. Approaches [1, 14] utilized extra labels to mitigate bias in feature representations, such as removing gender bias from age classification. While we focus on multiple factors, especially those rarely explored, and make use of parameters in synthetic data and data augmentation, which are cheaper labels compared with other work.

Action Recognition. Action recognition model robustness to nuisance factors has gradually received more attention in recent years and is found to be fragile. Lyu *et al.* [21] showed TSN [37] and I3D [4] make predictions relying on irrelevant information, such as related objects. Li *et al.* [17] revealed that current action datasets are biased towards the scene, objects or people. Thus our work focuses on RGB-based models and tries to increase their robustness even learning from a biased dataset. Choi *et al.* [6] introduce two adversarial losses to remove background effect while our work focus on multiple nuisance factors, including both 2D image corruptions and 3D factors e.g. viewpoint.

Usage of extra labels. Intermediate supervision and multi-task focus on the usage of extra labels. Multi-task learning focuses on related tasks. e.g. pose estimation and action recognition [12], surface normals and depth [24], *etc*. While we use labels not specifically annotated for any tasks, but parameter labels with less semantic meanings. DSN [15] is the first to propose the concept of deep supervision where the final label is used for supervising intermediate layers in a network. Furthermore, Li *et al.* [16] utilized intermediate shape concepts acquired from rendered data to supervise CNN hidden layers and achieved synthetic to real generalization. Whereas nuisance labels used in our method is different from the final label or intermediate concepts, which are easy-acquired but often ignored.

Synthetic data in vision tasks. Synthetic data is widely used as augmentation and shows merits on vision tasks. The domain gap between synthetic and real is always a problem to solve and the various intermediate information during rendering is rarely utilized. Previously, domain randomization [33] is a popular way for bridging the real-sim gap. On the one hand, our proposed NLS helps reduce the domain gap and offers a good way for using extra information. On the other hand, synthetic data serves as a source of nuisance labels, providing a larger application space for our method, including action recognition [8, 39, 34], pose estimation [5, 26, 27], segmentation [35, 25, 28], etc.

3. Method

This section can be roughly divided into four parts. First, we formulate the problem of using nuisance labels to improve model robustness (Sec. 3.1). Second, we illustrate the structure and adversarial training procedure of our proposed NLS module (Sec. 3.2). Third, we show the generation of nuisance labels and synthetic data (Sec. 3.3). Finally, we define *Dependency Degree on Nuisance Factor* and study it with nuisance labels (Sec. 3.4).

3.1. Problem Statement

The objective of our work is to weaken the influence of nuisance factor perturbations by making full use of nuisance labels. We demonstrate our method on action recognition tasks because the action model is easily affected by various factors.

We aim to train a model robust to nuisance factor perturbations, especially when training and testing data have different nuisance factor values. We suppose the process of making prediction y is written as:

$$y = F(x) = F(D(\phi, \psi)) \tag{1}$$

where F denotes the model, y denotes model prediction, the factors for generating input $x \in X$ can be divided into essential factors $\phi \in \Phi$ and nuisance factors $\psi \in \Psi$, D represents data generator. Thus an ideal model robust to nuisance factors should meet the following characteristics:

$$\forall \psi_i, \psi_j \in \Psi, F(D(\phi, \psi_i)) = F(D(\phi, \psi_j)) \tag{2}$$

where no matter what value nuisance factors take, as long as essential factors do not change, final prediction should stay the same.

In order to better illustrate the problem, we give the definition of important terms. Examples specific to action recognition are given after the definition of each term.

Nuisance Factors (Ψ): factors irrelevant to final prediction. The values of nuisance factors are nuisance labels and can be acquired from the data generation procedure. In this work, we divide the nuisance labels into three types according to their resources: 1) image processing parameters; 2) metadata in the data collection procedure; 3) rendering parameters from synthetic data generation. The details are shown in Sec. 3.3

Essential Factors (Φ): factors decisive to final prediction. For action recognition, Φ is human motion, which is directly relevant to action definition.

Task Label (Y): label for the main task. Task label does not necessarily equal essential factor labels, it is decided by essential factor values. Task label is unique in a certain task while essential factors may be more than one. For example, the essential factor in action recognition is human motion, and the task label is an action class.

To achieve the goal, we want the extracted features to contain the least information about nuisance factors. Most classification model F can be divided into feature extractor f with parameter θ_f and classifier g with parameter θ_g. In this way, the objective can be defined as:

$$\min_{\theta_f} I(\psi; f(x; \theta_f)) \tag{3}$$

where input data $x \in X$, nuisance factor $\psi \in \Psi$, $I(a; b)$ denotes the mutual information between a and b.

The intuition behind minimizing mutual information between ψ and $f(x)$ is that the main task classifier g takes extracted features $f(x)$ as input, so if $f(x)$ contains no information about nuisance factors, g will not utilize nuisance factor features, thus being robust to nuisance factor perturbations.

3.2. Nuisance-label Supervision (NLS) Module

In this section, we present the formulation of the proposed nuisance-label supervision module. The module utilizes nuisance labels as extra supervision, in order to train a feature representation with the least information about nuisance factors.

Overview. We give an overview of the whole network architecture in Fig. 2. We utilize adversarial training to achieve the goal of ”removing nuisance information” inspired by [10]. We divide input data into two parts: 1) data w/o nuisance labels, 2) data w/ nuisance labels. On the one hand, both inputs are put through a feature extractor and classifier the same as normal action recognition pipeline. On the other hand, data with nuisance labels is additionally put into NLS module for training feature representation. In this process, nuisance labels are used as extra supervision signals and output an adversarial loss.

Architecture. The NLS module consists of two parts: gradient reversal layer [10] and nuisance factor classifier as is shown in Fig. 3. The module input is extracted features and nuisance labels. The nuisance factor classifier is to predict labels of a certain nuisance factor, while corresponding nuisance labels are used for calculating a cross-entropy loss, which is widely used for classification tasks. The gradient reversal layer multiplies the gradient by a negative constant during the back-propagation process, making the input features as indistinguishable as possible for the nuisance factor classifier and ensuring the training is in an adversary way.

$$L_\psi = \mathbb{E}_{(x,\psi)\sim(X,\Psi)} L_{ce}(h(f(x; \theta_f); \theta_h), \psi) \tag{4}$$

where L_{ce} denotes cross-entropy loss.

Usually, there is more than one nuisance factor in a task, so we use multiple nuisance factor classifiers (h_p) to deal

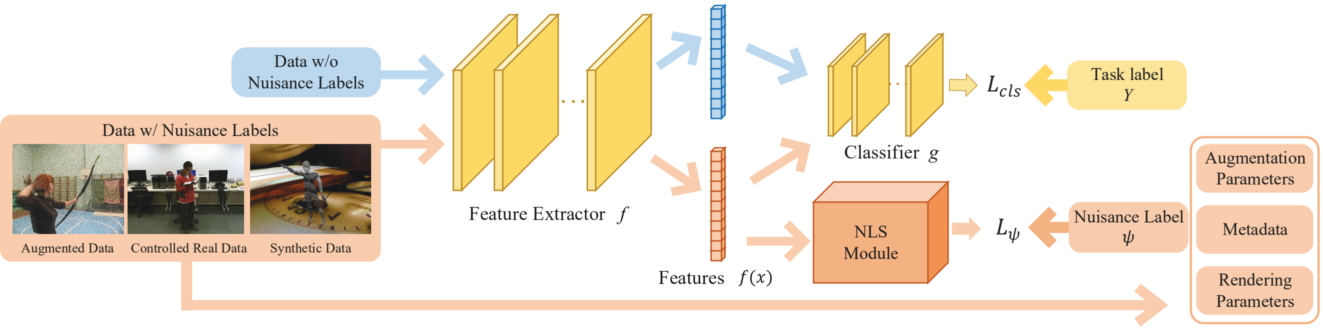

Figure 2. The pipeline of the proposed method. The blue arrow represents data w/o nuisance labels while the red arrow represents data w/ nuisance labels. Both inputs are put through feature extractor f and classifier g as yellow arrow shows. Additionally, data with nuisance labels is put into NLS module for training feature representations. (Better view in color)

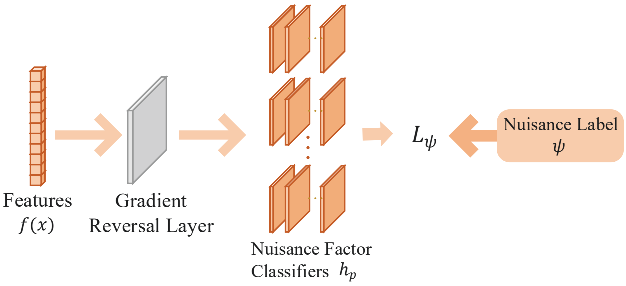

Figure 3. The architecture of Nuisance-label Supervision Module. The module consists of gradient reversal layer and nuisance factor classifiers. The module input is extracted features and nuisance labels. And the module is supervised by the nuisance adversarial loss L_ψ.

with each nuisance factor, then aggregate the outputs. The nuisance adversarial loss for multiple factors is defined as: $L_\psi = \sum_{p=1}^{N} L_{\psi^p}$, where superscript p denotes the p^{th} type of nuisance factors, and N denotes the number of nuisance factors matter in the task. Fig. 3 also illustrates how we deal with multiple nuisance factors in NLS module.

Training Procedure. Additional to removing nuisance factor effects, we certainly need the model to learn useful features for the main classification task:

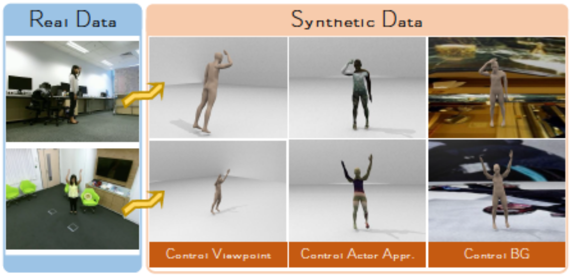

Figure 5. Examples of synthetic data. The appearance is similar in each row and different in each column. But the action label of each row is the same. So a robust model needs to be invariant to these dramatic nuisance factors change. We utilize nuisance labels from synthetic data to achieve this goal.

can save the kernel size, these can both be used as nuisance labels.

Similarly, some existing real datasets also contain such information, i.e. metadata during data collection. People record metadata only for evaluation or showing randomization, while it can be used in training as well. For example, people collect NTU RGB-D dataset in a controlled way, that for each video, the recording environment, subject appearance, and camera viewpoint are all saved. This information can be regarded as nuisance labels.

Synthetic data generation offers a convenient way to manipulate multiple factors in the virtual world and we can easily reach the ground truth of these factors. In our work, we use CG renderer Blender[1] to manipulate viewpoint, textures in the virtual world, as is shown in Fig. 4 (more details are in Appendix). We save camera position, actor appearance id and background id as nuisance labels. The examples of our CG synthetic data are shown in Fig. 5.

3.4. Dependency Degree on Nuisance Factor

Besides serving as extra supervision signals, nuisance labels are also used to probe model dependency degree on certain factors.

We design *Dependency Degree on Nuisance Factor* to quantitatively measure how much a trained feature extractor focuses on nuisance factors. We attach a classifier to a trained feature extractor to predict the correct label of a certain nuisance factor. During training, weights in the feature extractor are frozen while parameters in the attached classifier are optimized. For a classification task, the worst performance corresponds to the random guess of inputs to classes, i.e. chance level performance; the chance level performance on a nuisance factor ψ is denoted as $Acc_{Rand}(\psi)$. Thus we define the model dependency degree on a certain nuisance factor ψ based on trained feature extractor f as:

pearance, and recording environment. The metadata can be used as nuisance labels in NLS module. In addition, with these annotations, we create a more challenging data split, denoted as Cross-Nuisance (CN) split, where training and testing sets share no overlap on these three factors (details shown in Appendix). This split is for evaluating model robustness to 3D nuisance factor perturbations.

4.2. Implementation Details

Network architecture. For MNIST-C, we use the same architecture as in [22]. For action recognition, we choose I3D [4] network as the baseline. We initialize the network with inflated weights from the 2D Inception network pretrained on ImageNet [9]. At training time, we randomly sample 16 consecutive frames temporally, rescale the shorter side of all input frames to 226 pixels and randomly crop a 224×224 patch from each frame. The left-right flip is randomly applied as 2D data augmentation. At test time, we temporally sample three fragments evenly from the full video.

Our proposed NLS module is applied to the input feature map of the main task classifier. We choose $\alpha = 0.5$ for gradient reversal layer [10], which is the coefficient for reversed gradient during backpropagation. The nuisance classifier in NLS module is a three-layer MLP, with 1024 nodes in each layer.

Training Details. For MNIST-C, we train the network following the same rule as in [29]. For action recognition, we set batch size as 8 on each GPU. We use SGD optimizer with momentum = 0.9, weight decay = 1e-4. The initial learning rate equals 0.01 and decays to 10% when accuracy on validate set saturates. In order to prevent NLS from back-propagating noisy signals at the beginning of training, we set λ=0 for 10 epochs, then increase it to 0.05 for the rest of the training. We set λ so small because we want L_ψ to work as a regularization term, not to interfere with the main task training.

Synthetic Data Generation. When generating synthetic data for NTU RGB-D, animation data captured from Microsoft Kinect [40] are used to drive 3D human skeletal meshes from MakeHuman[2]. Following the concept of domain randomization [33], the textures for human appearance and background are randomly sampled from 1000 images within the MSCOCO dataset [18]; the camera is randomly set on 240 positions to record videos.

4.3. NLS on Image Processing Parameters

In this section, we use corrupted data as augmentation and regard its processing parameters as nuisance labels. We regard GNT and ANT1$\times$1 in [29] as the comparison, where Gaussian Noise and adversarial noise is added to part of the training data respectively as data augmentation.

Model	Clean	MNIST-C
Baseline	99.13	86.86
GNT	99.40	92.39
ANT1$\times$1	99.37	92.33
GNT+NLS	**99.44**	**92.51**

Table 1. Accuracy on MNIST (clean data) and MNIST-C (corrupted data) test sets. We compare our NLS with GNT and ANT1$\times$1, where models trained with noise and our NLS module.

MNIST-C Experiments. We use experiments on MNIST-C [23] to evaluate models on image corruptions. MNIST-C is a simple but popular dataset for people to study image corruption. Our proposed method NLS module is applied to GNT, where we regard Gaussian Noise standard deviations as nuisance labels. The results are shown in Table 1. Our method surpasses GNT and ANT on both clean data and corrupted data.

UCF101 Experiments. We further evaluate NLS module dealing with image corruptions on UCF101 dataset. We build a corrupted test set to evaluate the model robustness on image corruptions, as shown in Fig. 6. For each corruption, we study three types. As for the data augmentation baseline *Aug*, we choose only one for augmentation during training. During testing, all the types are applied respectively. We add one nuisance factor classifier for each type, through which only augmented data with corresponding corruption will pass. For comparison, we train an ANT1 $\times$ 1 and apply adversarial noise to each frame in a video sequence.

The results are shown in Table 2. We display accuracy on the clean test set and our designed corrupted test sets. The number is the mean value on three splits of UCF101. *Aug* has a good improvement on the corrupted test set especially on noise compared to the baseline but brings harm to clean data. Adding NLS further lifts model accuracy on clean data by 0.43% and corrupted data by 0.55% while adding adversarial noise to video data has marginal improvement. Though in some cases NLS is not the best, we highlight the consistent improvement compared with *Aug*.

4.4. NLS on Real Metadata

In this section, we evaluate NLS on real metadata in NTU RGB-D dataset, using its metadata, including camera viewpoint, actor appearance, and recording environment, as nuisance labels.

We conduct experiments on CN split (Sec. 4.1) and study three factors: viewpoint, actor appearance, and background in action recognition tasks. We study both real metadata and rendering parameters on the same dataset NTU RGB-D, so we denote adding NLS on real metadata as *Real NLS* and adding NLS on synthetic rendering parameters as *Sim NLS* (shown in Sec. 4.5).

The results of Real NLS are shown in the top two rows in

[2] http://www.makehumancommunity.org/

Model	Clean	Mean	Noise			Color			Blur		
			Gauss.	Salt.	Lap.	Gray	MulHue.	Contra.	Gauss.	Med.	Mot.
Baseline	**93.41**	77.85	63.17	45.15	47.65	90.58	90.57	**92.62**	92.53	87.80	90.60
Aug	92.30	90.18	90.36	87.17	89.07	91.58	89.69	91.43	92.00	89.61	90.70
ANT1×1	93.10	78.56	67.36	47.63	48.50	90.68	**90.72**	92.07	92.04	87.50	90.57
Aug+NLS	92.73	**90.73**	**91.07**	**87.79**	**89.58**	**91.90**	90.31	91.57	**92.57**	**90.64**	**91.10**

Table 2. Top-1 accuracy on UCF101 original and corrupted test sets. *Aug* refers to data augmentation that three types of image corruptions are used as data augmentation during training.

Model	Accuracy	F1 Score
Baseline	64.50	66.92
Real NLS	65.57	67.32
Sim Aug	68.75	70.99
Sim Aug+ANT1 × 1	68.04	71.05
Sim Aug+Sim NLS	70.40	73.03
Sim Aug+Real&Sim NLS	**70.64**	**73.38**

Table 3. I3D accuracy and F1 score on NTU RGB-D Cross-Nuisance (CN) split. *Real NLS* refers to applying NLS to real metadata and *Sim NLS* refers to applying NLS to synthetic rendering parameters. *Sim Aug* refers to using CG synthetic data as data augmentation.

Method	Modality	CS	CV
Hands attention [2]	RGB+Skeleton	84.8	90.6
DA-Net [36]	RGB+Flow	88.1	92.0
Pose evolution [19]	RGB+Depth	91.7	95.3
Hands attention [2]	RGB	75.6	80.5
Pose evolution [19]	RGB	78.8	84.2
Multi-task [20]	RGB	85.5	-
Glimpse clouds [3]	RGB	86.6	93.2
I3D [4]	RGB	90.2	95.2
I3D + Real NLS	RGB	**90.7**	**95.6**

Table 4. Comparison with state-of-the-art on standard splits of NTU RGB-D dataset. Our method achieves the highest performance within all the approaches using only RGB input.

Table 3. Adding Real NLS brings an accuracy improvement of 1.07% compared with the baseline model. The accuracy improvement is totally for free because no extra data is used.

Comparison with State-of-the-art. We conduct experiments on NTU RGB-D standard CV and CS splits, and compare it with state-of-the-art methods using *RGB only*. Note the data we use is totally the same as that in other methods. The results are shown in Table 4. Based on the strong baseline I3D, our proposed method reaches the best performance within RGB-based models, bringing accuracy improvement by 0.5% and 0.4% on two splits respectively. Besides, comparing with the results on CN split in Table 3, we observe that when the split is harder, the improvement brought by NLS is larger.

Model	Real Accuracy
DR	18.16
DR + NLS	**20.40**

Table 5. Top-1 accuracy on NTU RGB-D real test set of I3D trained on domain randomization (DR) synthetic data w/ and w/o adding NLS. We train models purely on synthetic data to show that NLS has ability of reducing domain gap.

4.5. NLS on Rendering Parameters

In this section, we evaluate NLS on rendering parameters, denoted as *Sim NLS*. The real dataset setup is the same as in Sec. 4.4. We generate CG synthetic data following rules in Sec. 3.3 as data augmentation, randomizing viewpoint, actor and background respectively, denoted as *Sim Aug*. We add ANT1×1 noise to synthetic training data as a comparison.

The results of Sim NLS are shown in Table 3. Adding CG synthetic data into training brings an accuracy improvement of 4.25% while using Sim NLS further lifts the performance by 1.65%. We notice ANT1×1 has no improvement for 3D factor robustness, which shows the limitation of current 2D augmentation methods. We also study the effectiveness of adding Real and Sim NLS simultaneously, which reaches the best performance on CN test set.

Domain Generalization. We explore NLS module potential in domain generalization. The models are trained on *pure* synthetic data and tested on real data. The synthetic data is generated following the rule of domain randomization [33].

The results of adding NLS on viewpoint, actor appearance, and background together are shown in Table 5. After adding NLS on synthetic rendering parameters, the model accuracy on real data improves observably by 2.24%.

4.6. Ablation Study

4.6.1 NLS on single type of corruption

Besides applying NLS on multiple nuisance factors, we study the performance improvement on a single factor. We apply NLS to single image corruption on UCF101 dataset. We augment training data by one type of image augmentation and add the corresponding NLS.

Model	Gauss.	Salt.	Lap.
Baseline	63.17	45.15	47.65
+Noise Aug	90.28	87.70	89.24
+Noise Aug+NLS	**90.63**	**88.27**	**89.59**
	Gray	MulHue.	Contra.
Baseline	90.58	90.57	92.62
+Color Aug	**92.57**	91.00	**92.14**
+Color Aug+NLS	**92.57**	**91.16**	92.13
	Gauss.	Med.	Mot.
Baseline	91.21	90.60	87.80
+Blur Aug	92.16	90.97	88.76
+Blur Aug+NLS	**92.61**	**91.50**	**88.94**

Table 6. Model accuracy on each corruption type in UCF101 test set. Only one type of corruption and its corresponding NLS are added to each model.

The results in Table 6 show that NLS on a single nuisance factor also improves model robustness. No matter NLS is applied to which factor, the performance is better than only using augmentation. One interesting phenomenon is that NLS brings an even larger lift to unseen corruptions, such as Salt and Pepper Noise and Median Blur, showing the ability for domain generalization.

4.7. Whether NLS helps reduce over-fitting

We design experiments to study whether NLS helps reduce over-fitting on the corresponding nuisance factor using NTU RGB-D dataset. In CN test set, all of the three factors are changed, so it is hard to say whether all the factors benefit from NLS. We apply NLS to one factor each time and evaluate it on three special test sets: *View Diff*, *BG Diff* and *Actor Diff*. These sets have only **one** factor different from training data, e.g. *View Diff* has the same actor and background settings but different viewpoint settings. We regard the performance improvement on *Diff* test set as nuisance factor over-fitting reduction.

Table 7 shows the results. After adding NLS, the corresponding factor over-fitting reduces obviously. Adding background NLS brings the smallest improvement, we explain it as the background in NTU RGB-D is less distinguishable than other factors. In addition, adding NLS on one nuisance factor also benefits robustness to other factors, e.g. accuracy on View Diff and BG Diff lifts sharply when adding NLS only on actor. While in most cases, adding NLS on three nuisance factors reaches the best performance.

4.8. How much does NLS reduce model dependency on nuisance factors

We use nuisance labels to diagnose the model dependency degree on each 3D nuisance factor, further explaining the reason for robustness improvement. As defined in Sec. 3.4, we train the attached nuisance factor classifier with feature extractor weights frozen. Fig. 7 shows the dependency degree changes, all the results are trained and tested on synthetic data. After adding NLS, the corresponding nuisance factor dependency degree decreases sharply.

Model	View Diff	BG Diff	Actor Diff
Baseline	70.96	87.59	87.67
+Sim Aug	73.58	88.46	88.06
+Sim Aug+V NLS	75.1	88.59	88.12
+Sim Aug+B NLS	74.43	88.89	87.71
+Sim Aug+A NLS	74.78	89.15	**88.63**
+Sim Aug+3F NLS	**75.74**	**89.96**	88.31

Table 7. Top-1 accuracy on special test sets, the higher accuracy, the less over-fitting to a certain nuisance factor. *View Diff* has only different viewpoint settings from training set. 3F refers to adding NLS on three nuisance factors and V, B, A refer to adding NLS on viewpoint, background, actor respectively.

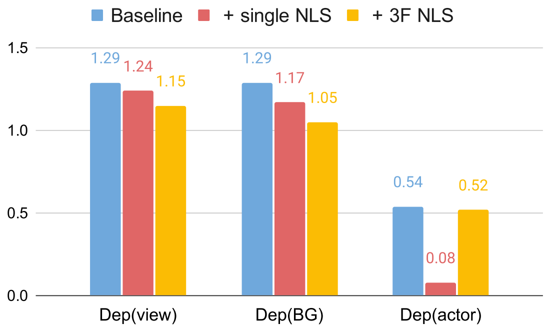

Figure 7. Model dependency degree on three 3D nuisance factors w/ and w/o NLS. The random guess for view, BG and actor is 0.42%, 0.1% and 0.1% respectively.

5. Conclusion

In this paper, we focus on the robustness problem caused by spurious correlation. We propose a novel module NLS using nuisance labels as extra supervision to break up spurious correlations, and we give three low-cost ways of acquiring nuisance labels, including image processing parameters, real metadata, and rendering parameters. Our proposed module effectively improves model robustness to nuisance factor perturbations. We demonstrate its effectiveness on MNIST-C, UCF101 and NTU RGB-D, handling image corruption and appearance changes, where our method all reaches the best performance. Furthermore, we utilize nuisance labels to diagnose the model dependency degree on a certain nuisance factor, explaining the reason why NLS helps reduce over-fitting. We hope that this work can contribute to the study of nuisance labels usage and enable more robust models.

References

[1] Mohsan Alvi, Andrew Zisserman, and Christoffer Nellåker. Turning a blind eye: Explicit removal of biases and variation from deep neural network embeddings. In *ECCV*, pages 0–0, 2018.

[2] Fabien Baradel, Christian Wolf, and Julien Mille. Pose-conditioned spatio-temporal attention for human action recognition. *CoRR*, abs/1703.10106, 2017.

[3] Fabien Baradel, Christian Wolf, Julien Mille, and Graham W. Taylor. Glimpse clouds: Human activity recognition from unstructured feature points. In *CVPR*, 2018.

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[5] Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Zhenhua Wang, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Synthesizing training images for boosting human 3d pose estimation. In *3DV*. IEEE, 2016.

[6] Jinwoo Choi, Chen Gao, Joseph CE Messou, and Jia-Bin Huang. Why can't i dance in the mall? learning to mitigate scene bias in action recognition. In *NeurIPS*, pages 851–863, 2019.

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, pages 113–123, 2019.

[8] César Roberto de Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel López. Procedural generation of videos to train deep action recognition networks. In *CVPR*, 2017.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[11] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[12] Georgia Gkioxari, Bharath Hariharan, Ross Girshick, and Jitendra Malik. R-cnns for pose estimation and action detection. *arXiv preprint arXiv:1406.5212*, 2014.

[13] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[14] Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning not to learn: Training deep neural networks with biased data. In *CVPR*, pages 9012–9020, 2019.

[15] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570, 2015.

[16] Chi Li, M Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D Hager, and Manmohan Chandraker. Deep supervision with shape concepts for occlusion-aware 3d object parsing. In *CVPR*, pages 5465–5474, 2017.

[17] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *ECCV*, pages 513–528, 2018.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[19] Mengyuan Liu and Junsong Yuan. Recognizing human actions as the evolution of pose estimation maps. In *CVPR*, 2018.

[20] Diogo C. Luvizon, David Picard, and Hedi Tabia. 2D/3D pose estimation and action recognition using multitask deep learning. In *CVPR*, 2018.

[21] Jialing Lyu, Weichao Qiu, and Alan Yuille. Identity preserve transform: Understand what activity classification models have learnt. In *CVPRW*, pages 8–9, 2020.

[22] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[23] Norman Mu and Justin Gilmer. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

[24] Zhongzheng Ren and Yong Jae Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 762–771, 2018.

[25] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016.

[26] Grégory Rogez and Cordelia Schmid. Mocap-guided data augmentation for 3d pose estimation in the wild. In *NeurIPS*, 2016.

[27] Grégory Rogez and Cordelia Schmid. Image-based synthesis for deep 3d human pose estimation. *IJCV*, 2018.

[28] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[29] Evgenia Rusak, Lukas Schott, Roland Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. Increasing the robustness of dnns against image corruptions by playing the game of noise. *arXiv preprint arXiv:2001.06057*, 2020.

[30] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *CVPR*, pages 1010–1019, 2016.

[31] Krishna Kumar Singh, Dhruv Mahajan, Kristen Grauman, Yong Jae Lee, Matt Feiszli, and Deepti Ghadiyaram. Don't judge an object by its context: Learning to overcome contextual bias. In *CVPR*, pages 11070–11078, 2020.

[32] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[33] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pages 23–30. IEEE, 2017.

[34] Gül Varol, Ivan Laptev, Cordelia Schmid, and Andrew Zisserman. Synthetic humans for action recognition from unseen viewpoints. *arXiv preprint arXiv:1912.04070*, 2019.

[35] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, pages 109–117, 2017.

[36] Dongang Wang, Wanli Ouyang, Wen Li, and Dong Xu. Dividing and aggregating network for multi-view action recognition. In *ECCV*, 2018.

[37] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[38] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, pages 3712–3722, 2018.

[39] Yi Zhang, Xinyue Wei, Weichao Qiu, Zihao Xiao, Gregory D Hager, and Alan Yuille. Rsa: Randomized simulation as augmentation for robust human action recognition. *arXiv preprint arXiv:1912.01180*, 2019.

[40] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[41] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016.