applied sciences

MDPI

Article

Learning Class-Specific Features with Class Regularization for Videos

Alexandros Stergiou *, Ronald Poppe and Remco C. Veltkamp

Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands; r.w.poppe@uu.nl (R.P.); r.c.veltkamp@uu.nl (R.C.V.)

* Correspondence: a.g.stergiou@uu.nl

Received: 28 July 2020; Accepted: 4 September 2020; Published: 8 September 2020

Abstract: One of the main principles of Deep Convolutional Neural Networks (CNNs) is the extraction of useful features through a hierarchy of kernels operations. The kernels are not explicitly tailored to address specific target classes but are rather optimized as general feature extractors. Distinction between classes is typically left until the very last fully-connected layers. Consequently, variances between classes that are relatively similar are treated the same way as variations between classes that exhibit great dissimilarities. In order to directly address this problem, we introduce *Class Regularization*, a novel method that can regularize feature map activations based on the classes of the examples used. Essentially, we amplify or suppress activations based on an educated guess of the given class. We can apply this step to each minibatch of activation maps, at different depths in the network. We demonstrate that this improves feature search during training, leading to systematic improvement gains on the *Kinetics*, *UCF-101*, and *HMDB-51* datasets. Moreover, *Class Regularization* establishes an explicit correlation between features and class, which makes it a perfect tool to visualize class-specific features at various network depths.

Keywords: class regularization; 3D-CNN; spatiotemporal activations; class-specific features

1. Introduction

Video-based action recognition has seen tremendous progress since the introduction of Convolutional Neural Networks (CNNs) [1,2]. The hierarchical application of 3D convolutional operations has been shown to effectively capture descriptive spatiotemporal features.

CNNs include multiple layers that are stacked in a single, hierarchical architecture. Features are calculated by successive convolutions. Kernels in early layers focus on simple textures and patterns, while deeper layers focus on more complex parts of objects or scenes. As these features become more dependent on the different weighting of neural connections in previous layers, only a portion of them becomes descriptive for a specific class [3,4]. Yet, all kernels are learned in a class-agnostic way. Consequently, much of the discriminative nature of CNNs is achieved only in the very last fully-connected layers. This hinders easy interpretation of the part of the network that is informative for a specific class.

We aim at forcing the network to propagate class-specific activations throughout the network. We propose a method named *Class Regularization* that relates class information to extracted features of network blocks. This information is added back to the network by amplifying and suppressing activation values with respect to predicted classes. *Class Regularization* has a beneficial effect on the nonlinearities of the network by modulating the effects of the activations. Owing to this, the architecture can effectively distinguish between the most class-informative kernels in each part of the network given a selected class. This procedure reduces the dependency on many uncorrelated features

in the last fully-connected layers that are responsible for the final class predictions, essentially penalizing overfitting.

Our contributions are the following:

- We propose *Class Regularization*, a regularization method applied in spatiotemporal CNNs. The method does not change the overall structure of the architecture, but can be used as an additional step after each operation or block.
- We demonstrate that the relationships between classes and features can be visualized by propagating class-based feature information through normalized neural weights for each of the model's building blocks.
- We report performance gains for benchmark action recognition datasets *Kinetics*, *UCF-101*, and *HMDB-51* by including *Class Regularization* in convolution blocks.

We discuss advances in vision-based action recognition in Section 2. A detailed overview of *Class Regularization* appears in Section 3. Experiments are presented in Section 4. We conclude in Section 5.

2. Related Work

Significant advancements have been made in the recognition of actions in videos with the introduction of deep neural approaches that are based on the hierarchical discovery of informative features [5,6]. These architectures provide the basis to further accommodate temporal information.

Due to the indirect relationship between temporal and spatial information, one of the first attempts on video recognition with neural models was the use of Two-stream networks [7]. These networks contain two separate models that use still video frames and optical flow as inputs. Class predictions are made after combining the extracted features of the separate networks. Two-stream networks were also used as a base method for works such as Temporal Segment Networks (TSN, [8]) which use scattered snippets from the video and fuse their predictions. This approach sparked research on the selection of informative frames [9,10]. Other extensions of Two-stream networks include the use of residual connections [11,12] that share spatiotemporal information across multiple layers.

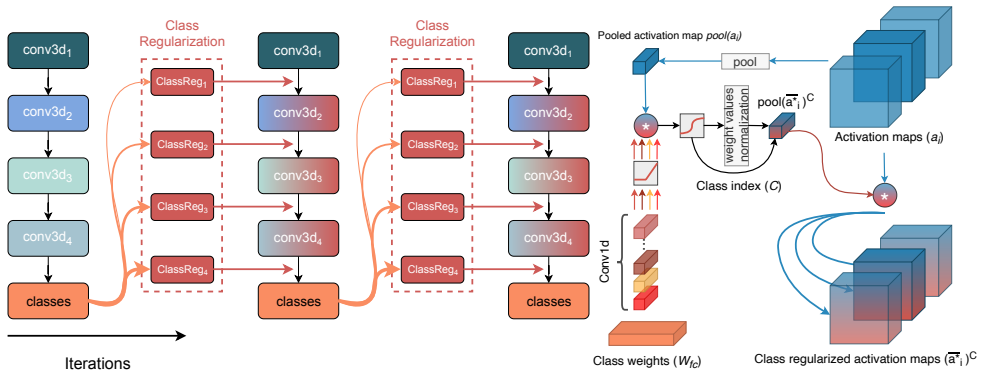
An alternative approach to capture temporal information in CNNs is through 3D convolutions [13]. 3D convolutions include an additional time dimension to the two spatial dimensions. 3D convolutions have been shown to outperform standard image-based networks [14] in video classification. A fusion of Two-stream networks and 3D convolutions has been explored with the I3D architecture [15]. The two spatiotemporal models are trained in parallel on frame and optical flow data, with the benefit of also processing temporal-only information. Further structures that have been explored with 3D convolutions include Residual Networks [16].

Inspired by depthwise and pointwise convolutions performed over image channels [17], researchers have introduced ways of splitting the 3D convolutions in subsequent 2D convolution operations. This can be achieved through either using spatial filters followed by temporal-only filters [18,19] or convolutions in groups [20,21]. Alternative implementations include the use of long-sequence and short-sequence kernels [22] and dimension-based iterations of time-width and time-height [23,24]. Others have worked on the minimization of the computational requirements by shifting activations along time [25]. Previous works that have touched upon regularization were aimed more at the overall temporal consistency of features (e.g., [26]) without the consideration of feature combinations that are most informative for some classes.

Although these techniques have shown great promise in terms of accuracy and computational performance, there is still a lack of better spatiotemporal representations for intermediate network layers. Currently, there is no standardized way of processing the temporal information. Our proposed method, named *Class Regularization*, fills this void as it can be added to virtually all network architectures with minimum additional computational cost in order to enhance the correlation between specific spatiotemporal features and the action class.

3. Regularization for Convolutional Blocks

In CNNs, explicitly adding class information through regularization is challenging, given the increasing level of ambiguity of the model's inner workings with respect to the network depth. We make the assumption that, when testing a video in a trained CNN, a speculative guess can be made on which class is represented by observing the activations produced at a certain layer. The underlying idea is that different kernels focus on different spatiotemporal patterns that appear in different classes. These guesses depend on the layer depth. Deeper network layers can distinguish class-specific features better because of the larger feature complexity. Therefore, guesses at different parts of the model should have consequently greater or lesser effect. To include the layer feature complexity, we define an *affection rate* ($\mathfrak{A}$) that specifies how the class predictions affect the network's activation maps. The values are chosen given the layer depth and the level of uncertainty of their class estimates. The discovery of feature correspondence between the layer's feature space and the class's feature space is implemented through pointwise convolutions and by incorporating their kernel updates as part of the training procedure. A complete workflow of the regularization method appears in Figure 1.



(**a**) Network view (**b**) Block view

Figure 1. Class Regularization. An additional pathway between class weights and intermediate features is added (**a**) connecting different parts of the network across iterations. In *Class Regularization* blocks, activation maps are pooled to vector volumes (pool(a_i)) and multiplied by the class weights in order to select the resulting highest class activation (pool$(\overline{a_i^*})^C$). A computational overview of the in-block operations (**b**) appears in Algorithm 1.

3.1. Layer Fusion with Class Predictions

We first discuss the main process for finding class estimates based on extracted features from the convolutional block at depth i in the network. We start by creating a vector representation of the features as distributed between the activation's channels for a spatiotemporal activation map input of size ($FxHxW$). Considering the produced activation map of the ith block (denoted as a_i), a global feature representation of the activations is created through a spatiotemporal sampling operation: $pool(a_i)$ (Equation (1)). The produced volume can be interpreted as a single vector descriptor containing a combination of the feature intensity values in the form of their average activations in a significantly lower dimensional space.

$$pool(a_i) = \frac{1}{F \times H \times W} \sum_{f=1}^{F} \sum_{h=1}^{H} \sum_{w=1}^{W} a_{(f,h,w,i)}. \quad (1)$$

To include class-based information in intermediate feature layers, we use the weights of the network's final prediction layer W_{fc}. This allows to establish a relationship between previous and current iterations in a recurrent fashion, by taking class-specific features into account. To obtain feature alignment given the channel sizes for the current activation and the prediction weights,

a one-dimensional convolutional operation ($conv = W_{fc} * \mathcal{W}$) is applied to the class weights with a kernel ($\mathcal{W}$) size of 1 followed by a *relu* activation function.

Next, we use a standard matrix-to-matrix multiplication to create an association between the channel descriptor and each of the classes. The volume Z_i is of size $\{Z_i^{[1]}, ..., Z_i^{[CL]}\}$, for CL classes, with each unit $Z_i^{[j]}$ having the same channel dimensionality as a_i. This operation allows an early estimate for the indexes of the most relevant features for each class.

One could use the outputs of the multiplication as a separate loss function but we have two main reasons for refraining from doing so. First, due to the limited feature complexity, it is significantly more difficult to make educated class predictions in early layers of the network. This also corresponds to the notion of hierarchical feature extraction in CNNs. Second, through these multiple output points, multiple error derivatives are to be calculated which will slow down the training process significantly.

3.2. Fusion of Class Weight Vector and Spatiotemporal Activation Maps

Our aim is to let the produced class-based activations Z_i become indicators for the most informative class features. With this, we aim at obtaining the maximum class probability through a standard softmax activation function applied on the activations. This converts the weighed sum logit score to a probabilistic distribution over all classes: $S(Z_i)$. Based on (5) in Algorithm 1, the index of the maximum class activation is selected (C) to define the class weight vector with the highest correspondence based on the class features of the layer.

Algorithm 1 Class regularization overview

1: **Inputs:**
Values of activation map a_i for ith layer.
2: **Outputs:**
Class-regularized activations $(\overline{a_i^*})^C$
3: $W_i \leftarrow relu(W_{fc} * \mathcal{W})$ ▷ Weight dimensionality conversion
4: $Z_i \leftarrow W_i * pool(a_i)$ ▷ Weighted sum per class neuron
5: $C \leftarrow \underset{j}{\text{argmax}}\{S(Z_i^{[1]}), ..., S(Z_i^{[CL]})\}\ \forall\ S(Z_i^{[j]}) = \frac{\epsilon^{Z_i^{[j]}}}{\sum_{c \in 1,...,CL} \epsilon^{Z_i^{[c]}}}$ ▷ Largest softmax activation class search
6: $\widehat{W}_i \leftarrow \mathfrak{A} * \frac{(W_i - min\{W_i\}) * (1 - \mathfrak{A})}{max\{W_i\} - min\{W_i\}}$ ▷ Weight scaling
7: $(\overline{a_i^*})^C \leftarrow \widehat{W}_i^{[c]} * a_i$ ▷ Final weight regularization.

As we want to amplify activation features, we need to normalize weight vector W_i before fusing it with the layer's activation maps. Layer features that are less informative for a specific class should be scaled down, while informative ones should be scaled up. To this end, we set a value ($\mathfrak{A}$) which will be referred to as the *affection rate*. It determines the bounds that the weight vector will be normalized to ($\widehat{W}_i$)—see step (6) in Algorithm 1. We are not using a standardization method as in *batch normalization* [27] that guarantees a zero-mean output. This is because we use a multiplication operation for including the class weight information to the activation maps. Therefore, zero-mean normalization will remove part of the information because values below one will decrease in the feature intensity. It also hinders performance as it effectively contributes to the occurrence of *vanishing gradients* because the produced activation map values would be reduced at each iteration.

In our final step, we inflate the normalized weight vector $\widehat{W}^{[c]}$ to match the dimensions of the spatiotemporal activation maps. We then perform a matrix-to-matrix multiplication between the

activation maps of the layer a_i and the normalized weights with the axis of symmetry being the depth or channel's dimension.

3.3. Performing Updates to Regularized Volumes

Class Regularization is performed on activation maps in the network to manipulate the activation values of the upcoming operations. We underline that the value of the *affection rate* $\mathfrak{A}$ used in the normalization can be trained through a separate objective function. In addition, our method is independent of the training iteration or layer number that it is applied to, and can process examples independently for online learning. Therefore, the discovery of class connections can also be performed in minibatches, to then return regularized volumes over single or multiple class weights.

The feature representation that is captured $(\overline{a_i^*})^C$ depends on the specific example a_i, the layer or block i of the architecture that the regularization was applied to, the chosen class C that was selected, and the affection rate $\mathfrak{A}$. The distribution inside $\widehat{W}_i^{[C]}$ has an expected value of 1 and a distribution of $2(1-\alpha)$. Due to the low computational overhead to back-propagate through the proposed method, the computation times are not significantly affected by including *Class Regularization* in a network.

3.4. Class Regularization for Visualizations

As *Class Regularization* is based on the injection of class-based information inside the feature-extraction process, a direct correlation between classes and features is made at each block in which the method is applied. Being able to represent the class features given a different feature space improves the overall explainability capabilities of the model. Through feature correlation, the method alleviates the curse of dimensionality problem of previous visualization methods that rely on back-propagating from the predictions to a particular layer [28]. Since the classes are represented in the same feature space as the activation maps of the block, we can discover regions in space and time that are informative over multiple network layers. To the best of our knowledge, this is the first method to visualize spatiotemporal class-specific features at each layer of the network.

By extending the proposed algorithm to include an adaptation of *Saliency Tubes* [29] in each block, we can create visual representations of the features with the highest activations per class. We create two visual examples of the class *bowling* in Kinetics-400 to demonstrate class activations in different layers of the network (Figure 2). As observed in the clip segment, in both cases, early layer features are significantly less deterministic of the class and mostly target the distinction between foreground and background. In later layers, the focus shifts from the actor to the background in the predictions layer of the first clip. Ball and bowling pins are present at a wallpaper, which demonstrates that a strong still-frame visual signal is favored in case the action or objects within the action are occluded. In the second clip, the main field of focus of the network in the final layers is towards the area between the actor's hand and the ball. We note that, by design, all network architectures used fuse all temporal activations to a single frame at their final convolution block, which only allows the visualization of spatial extension of the activations.

The amplification of layer features can also be visualized as in Figure 3. The top-3 kernels to be amplified for a baseball hit example correspond to spatiotemporal features such as the appearance of the bat, the field, and the movement of the bat during a swing. In addition, these amplifications are also propagated to deeper layers in the network through the connections of the most informative kernels.

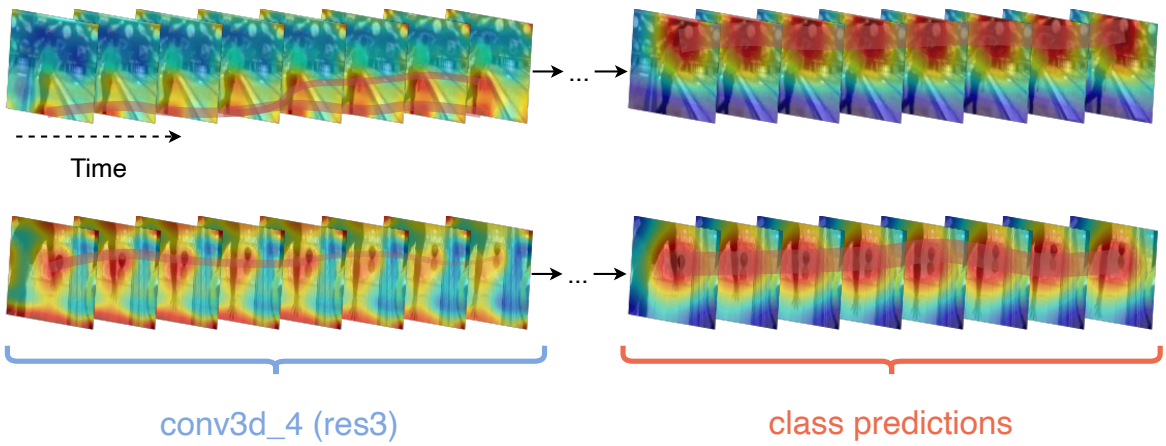


Figure 2. Layerwise class feature correspondence. Each of the *Saliency Tubes* [29] represents the class activations of a *Class Regularized* Wide-ResNet50 model in layers *res3* and *predictions (fc)*. Both clips correspond to visualizations for class *bowling* from Kinetics-400 [30] for two different examples with variations in their spatiotemporal regions.

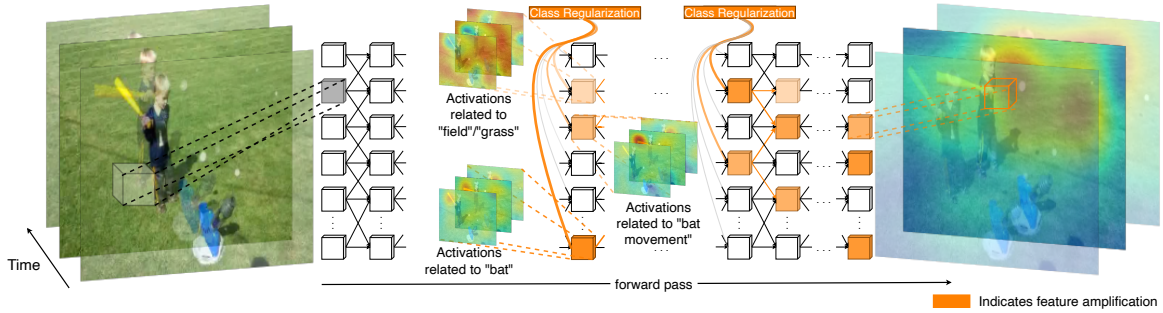


Figure 3. Visualization of feature amplification. As class-specific activations are reused by the network, informative spatiotemporal features for specific classes during an iteration are amplified. The effect of this amplification is propagated to deeper layers in the network through the connections of the layers in which *Class Regularization* is applied.

4. Experiments

We demonstrate the merits of *Class Regularization* on the action recognition classification performance on three benchmark datasets, and using a number of widely used CNN architectures. Results are summarized in Table 1. We further statistically compare the classification performance between predictions from different architectures and from different blocks within the same architecture.

For our experiments, we consider the widely used Kinetics-400 [30] dataset as a baseline. Then, each of the selected models is further fine-tuned on both UCF-101 [31] and HMDB-51 [32] by training the 1D convolutions inside *Class Regularization* to ensure a dimensionality correspondence between the new class weight vectors and the activation maps of each layer that the method is applied to. We further allow updates on the final two convolution blocks of the selected networks.

Training. The models trained on Kinetics are initialized with a standard Kaiming initialization [33] without inflating the 3D weights. This allows for a direct comparison between architectures with and without *Class Regularization* and to compare the respective accuracy rates. For all the experiments, we use an *SGD* optimizer with 0.9 momentum. *Class Regularization* is added at the end of each bottleneck block in the ResNets and at the end of each mixed block in I3D.

Table 1. Architectures of 3D convolution models with and without class regularization. We note that in the *Class Regularized* networks, only an eighth of the additional parameters are trainable, with the rest corresponding to nontrainable class weights tensor duplicates.

Layer Name	3D ResNet101 (w/o CN)	3D Wide ResNet 50 (w/o CN)	I3D (w/o CN)
$con3d_1$	$7 \times 7 \times 7, 64$ conv		
$con3d_2$	$\begin{bmatrix} 1\times1\times1 \\ 3\times3\times3 \\ 1\times1\times1 \end{bmatrix} (\times 64) \times 3$	$\begin{bmatrix} 1\times1\times1 \\ 3\times3\times3 \\ 1\times1\times1 \end{bmatrix} (\times 128) \times 3$	$\begin{bmatrix} 1\times1\times1 & 1\times1\times1 & 1\times1\times1 & 3\times3\times3, (pool) \\ & 3\times3\times3 & 3\times3\times3 & 1\times1\times1 \end{bmatrix} (\times 480) \times 2$
$ClassReg_1$	- / $\alpha = 0.9$	- / $\alpha = 0.9$	- / $\alpha = 0.8$
$con3d_2$	$\begin{bmatrix} 1\times1\times1 \\ 3\times3\times3 \\ 1\times1\times1 \end{bmatrix} (\times 128) \times 4$	$\begin{bmatrix} 1\times1\times1 \\ 3\times3\times3 \\ 1\times1\times1 \end{bmatrix} (\times 256) \times 4$	$\begin{bmatrix} 1\times1\times1 & 1\times1\times1 & 1\times1\times1 & 3\times3\times3, (pool) \\ & 3\times3\times3 & 3\times3\times3 & 1\times1\times1 \end{bmatrix} (\times 832) \times 5$
$ClassReg_2$	- / $\alpha = 0.8$	- / $\alpha = 0.8$	- / $\alpha = 0.7$
$con3d_2$	$\begin{bmatrix} 1\times1\times1 \\ 3\times3\times3 \\ 1\times1\times1 \end{bmatrix} (\times 256) \times 23$	$\begin{bmatrix} 1\times1\times1 \\ 3\times3\times3 \\ 1\times1\times1 \end{bmatrix} (\times 512) \times 6$	$\begin{bmatrix} 1\times1\times1 & 1\times1\times1 & 1\times1\times1 & 3\times3\times3, (pool) \\ & 3\times3\times3 & 3\times3\times3 & 1\times1\times1 \end{bmatrix} (\times 1024) \times 2$
$ClassReg_3$	- / $\alpha = 0.7$	- / $\alpha = 0.7$	- / $\alpha = 0.6$
$con3d_2$	$\begin{bmatrix} 1\times1\times1 \\ 3\times3\times3 \\ 1\times1\times1 \end{bmatrix} (\times 512) \times 3$	$\begin{bmatrix} 1\times1\times1 \\ 3\times3\times3 \\ 1\times1\times1 \end{bmatrix} (\times 1024) \times 3$	-
$ClassReg_4$	(-) / $\alpha = 0.6$	- / $\alpha = 0.6$	-
predictions	global average pool, softmax unit group		

We use transformations for both spatial and temporal dimensions. In the temporal dimension, we use clips of 16 frames that are randomly extracted from different subsequences of the video. For the validation sets, we only use the center 16 frames. Spatially, we use a cropping size of 112×112 and 256×256 for fine-tuning. All models are trained for 170 epochs, as no further improvements were observed afterwards in our initial experiments with all three networks on Kinetics. We also used a step-based learning rate reduction of 10% every 50 epochs.

Datasets. Kinetics-400 consists of roughly 240 K training videos and 20 k validation videos of 400 different human actions. We report the top-1 accuracy alongside the computational cost (FLOPs) for each of the networks using spatiotemporally cropped clips. UCF-101 and HMDB-51 have 13 k and 9 k videos, respectively. They are used to demonstrate the transfer abilities of the proposed *Class Regularization* as well as the usability of our method in smaller datasets.

4.1. Main Results

A comparison between results of models trained from scratch on Kinectics-400 appears in Table 2. Existing networks consider a complete change in the overall architecture or convolution operations in models, which is computationally challenging given the large memory requirements of spatiotemporal models. New models need to be trained for a significant number of iterations in order to achieve mild improvements, while additionally using large datasets [34,35] for pretraining. In contrast, the proposed *Class Regularization* method is used on top of existing architectures and only requires fine-tuning the dimensionality correspondence between the number of features in a specific layer and the features that are used for class predictions. Overall, the largest improvements were observed on networks with larger number of layers, such as ResNet101-3D, in comparison to networks with lower number of layers, with the best performing architectures being I3D and ResNet101-3D with *Class Regularization* with 67.8% top-1 accuracy and 67.7% top-1 accuracy, respectively.

Table 2. Accuracy rates of different spatiotemporal convolutional architectures on the Kinetics-400 dataset.

Model	Backbone	Depth	# Params (M)	GFLOPS	Top-1 (%)
ResNet50-3D [16]	ResNet	50	36.72	80.32	0.636
ResNet101-3D [16]	ResNet	101	69.06	110.98	0.652
ResNeXt101-3D [16]	ResNet	101	69.06	148.91	0.667
Wide ResNet50-3D [16]	ResNet	50	140.94	72.32	0.640
I3D [15]	Inception	48	12.07	55.79	0.664
R(2+1)D-ResNet50 [19]	Resnet	50	34.86	89.14	0.645
R(2+1)D-ResNet101 [19]	ResNet	101	67.22	159.21	0.668
MF-Net [36]	ResNet	50	8.03	22.7	0.653
ResNet101-3D w/*Class Regularization*	ResNet	101 + 4	37.10	126.13	0.677
Wide ResNet50-3D w/*Class Regularization*	ResNet	50 + 4	141.32	82.67	0.653
I3D w/*Class Regularization*	Inception	48 + 3	69.44	62.96	**0.678**

4.2. Direct Comparisons with and without Class Regularization

In Table 3, we pairwise compare three architectures after training directly on Kinetics, and after adding *Class Regularization* blocks and fine-tuning. For each architecture and dataset, networks with *Class Regularization* outperform those without. The largest gain is observed in the 101-layer 3D Resnet (+2.45%), while we obtained improvements of +1.37% and +1.43% on the Wide-ResNet50 and I3D, respectively. Our approach appears to be especially useful for deeper networks. Since the effective description of classes is achieved through large feature spaces, *Class Regularization* significantly benefits models that include complex and large class weight spaces. The set of influential class features can be better distinguished with minimal computational costs, as shown in Figure 4.

Table 3. Direct comparison of architectures with (in orange) and without (in black) *Class Regularization* blocks. Latency in msec.

Model	Added Latency	Kinetics	UCF101	HMDB51
ResNet101	-	65.29%	88.23%	62.47%
ResNet101	+ 98.786	67.74%	88.84%	63.31%
Wide ResNet50	-	63.96%	87.52%	61.62%
Wide ResNet50	+102.995	65.33%	89.11%	63.24%
I3D	-	66.42%	91.80%	64.27%
I3D	+68.34	67.85%	93.17%	65.83%

When transferring weights, the retraining process does not change for the *Class Regularization* networks as the additional training phase is only performed in order for the model to learn the feature correspondence.

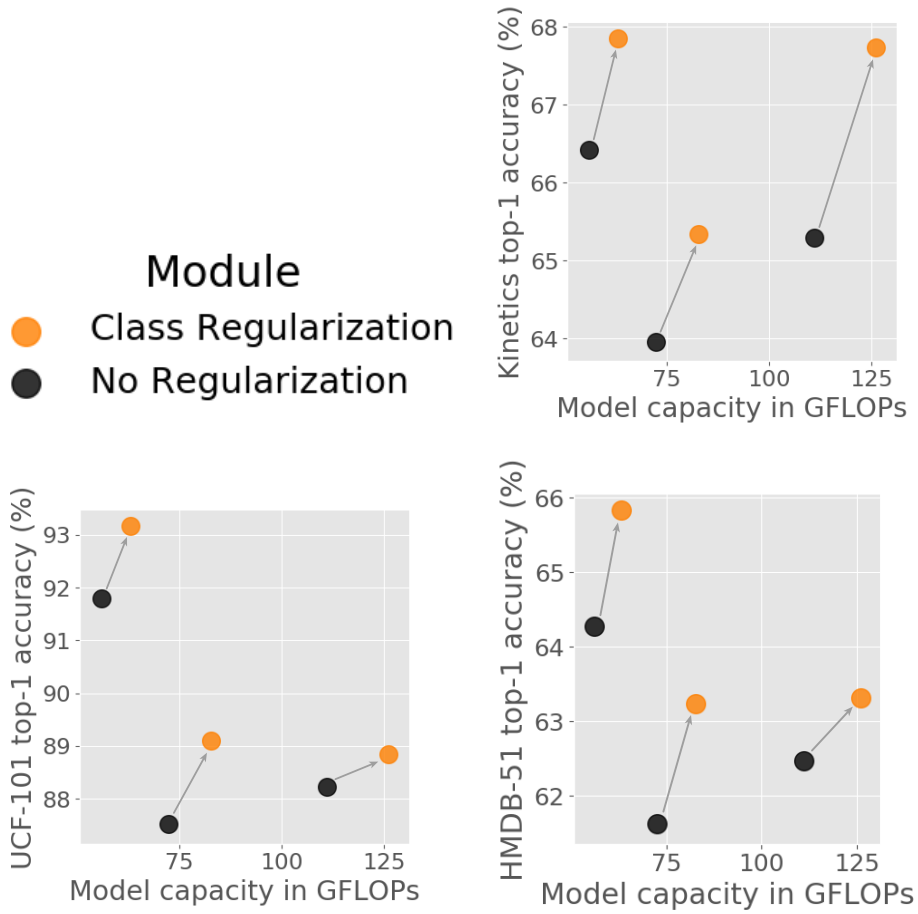


Figure 4. Class Regularization accuracy/computation trade-off.

Advancements can also be achieved in transfer learning, as shown by the rates for UCF-101 (split 1) and HMDB-51 (split 1) in Table 3. Accuracy for non-*Class-Regularized* networks are recalculated in order to ensure the same training setting. The largest gain from the original implementation in UCF-101 is on the Wide ResNet50 with +1.59% followed by I3D with +1.26% and ResNet101 +0.61%. For the HMDB-51 dataset, the model pair that exhibits the largest gap in performance is Wide ResNet50 with a +1.62% improvement, I3D with +1.56%, and ResNet101 with +0.84%. Overall, the minor deterioration of the accuracy gains in transfer learning could be contributed to the fact that kernels have been already trained in conjunction with class information from a different dataset.

As shown in Figure 5, *Class Regularization* demonstrates improvements for most classes in Kinetics. The greatest improvements are observed for classes that can be better defined based on their execution instead of their appearance. Examples are "bench pressing" (9.1% improvement), "high jump" (8.9% improvement), and "jogging" (15.6% improvement). Amplifying features that are characteristic of those classes improves the model's recognition capabilities. In contrast, classes that are more likely to contain significant variation in feature values are more prone to be classified wrongly. This is particularly true for classes that exhibit large intraclass variation, such as "parkour", where there are no standard actions performed. Other examples are "garbage collection" in Figure 5 which could be performed either mechanically (top), by a single person (mid), or by multiple people (bottom). Other examples include either oscillations as in "pumping gas", or require contextual information as in "sniffing".

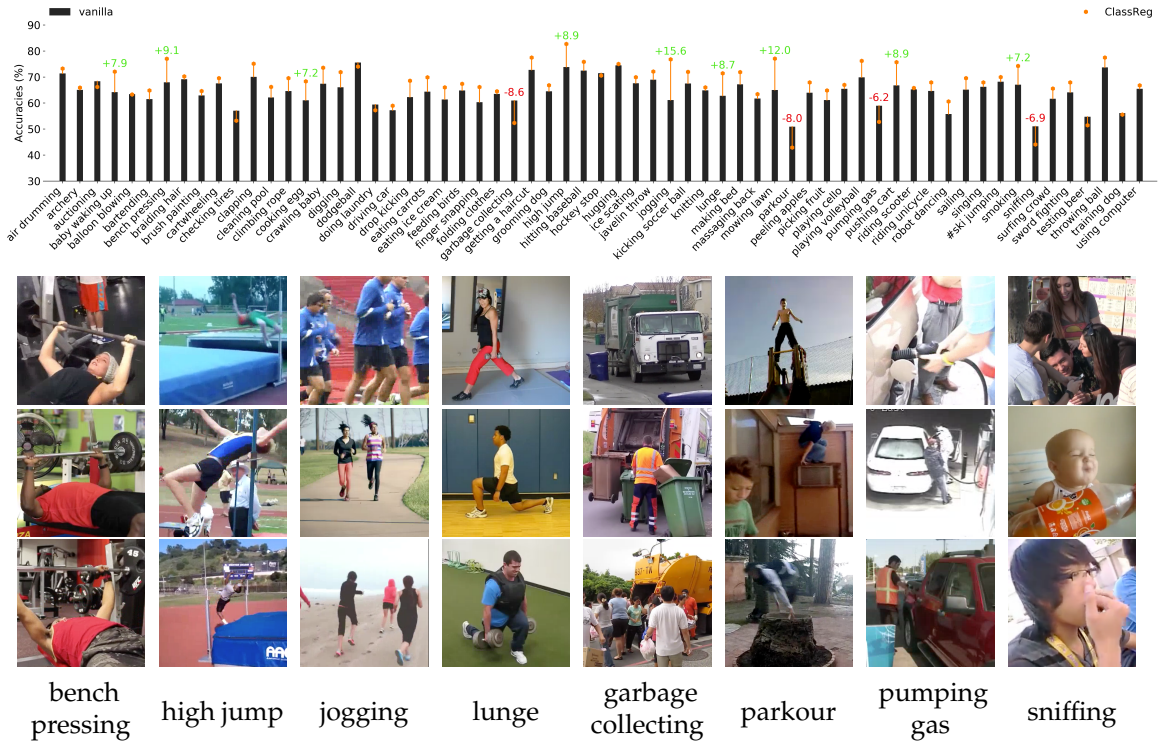


Figure 5. Per-class performance for ResNet-101 w/o Class Regularization with illustrative examples of classes with large performance gains or losses in Kinetics-400 [30].

4.3. Resulting Statistical Significance

To analyze whether the classification performance with the addition of *Class Regularization* to a network is statistically relevant, we perform a McNemar's test on UCF-101 and HMDB-51. We study if the difference in accuracy between a network with and without *Class Regularization* is to be attributed to *marginal homogeneity* (i.e., the result of sampling distribution) or if the variance is significant enough to conclude that the two models indeed perform differently. (Table 4a–c) summarize the McNemar tests, defined in Equation (2), where b are the cases in which the nonregularized model ($\neg Reg$) is correct (+) and the regularized (Reg) is wrong (-), and c are the cases in which the nonregularized model is correct while the regularized is wrong.

$$x^2 = \frac{(b-c)^2}{b+c} \quad \begin{cases} H_0 : P_b = P_c \\ H_1 : P_b \neq P_c \end{cases} \tag{2}$$

We notice that in each of the tested architectures, the c values were significantly larger than those of b. In particular, a standard 3D ResNet-101 model with class regularization is different than that of a model without, with x^2 values of 2.48 and 2.84 for UCF-101 and HMDB-51, respectively. For both datasets, the average probability that the difference between the model with and without is attributed to statistical error is approximately 10%. The difference is also evident in architectures that include a larger number of extracted features per layer. This can be seen from the results of the Wide-ResNet-50 model, in which the x^2 values are 3.40 and 3.08 for UCF-101 and HMDB-51, respectively. The difference is statistically significant at the $p = 0.05$ level. The final comparison considers I3D as the base network to also account for cases of convolution operations being split to multiple and differently sized cross-channeled convolutions. The probability that the measured difference is not due to a systematic difference in performance is approximately 10%. The sum of (b, c)

values in every case is reasonably large in order to sufficiently approximate x^2 and to conclude that marginal probabilities for models with and without *Class Regularization* are not homogeneous.

Table 4. McNemar's statistical significance test on the first split of UCF-101 and HMDB-51. Results with and without regularization for (**a**) ResNet101, (**b**) Wide-ResNet, and (**c**) I3D models. (**d**–**f**) present results for class-regularized ResNet101 across different blocks in the architecture.

a ResNet101-3D

	Reg (+)	*Reg* (-)
¬*Reg* (+)	773 / 924	23 / 35
¬*Reg* (-)	36 / 47	83 / 527

b Wide-ResNet50-3D

	Reg (+)	*Reg* (-)
¬*Reg* (+)	764 / 895	34 / 49
¬*Reg* (-)	51 / 68	65 / 518

c I3D

	Reg (+)	*Reg* (-)
¬*Reg* (+)	824 / 887	15 / 96
¬*Reg* (-)	27 / 118	51 / 429

d ResNet101-3D (over different depths)

	Reg^2 (+)	Reg^2 (-)
Reg^1 (+)	18 / 30	0 / 2
Reg^1 (-)	35 / 89	861 / 1411

e ResNet101-3D (over different depths)

	Reg^3 (+)	Reg^3 (-)
Reg^2 (+)	52 / 104	1 / 5
Reg^2 (-)	730 / 829	131 / 592

f ResNet101-3D (over different depths)

	Reg^4 (+)	Reg^4 (-)
Reg^3 (+)	772 / 930	10 / 3
Reg^3 (-)	37 / 38	95 / 559

Finally, we compare the class predictions found by the *Class Regularization* method in different blocks within the architecture in (Table 4d–f). These panels provide an overview of how different depths of the network perform for the given task by gradually ablating blocks of convolutions. As observed for both datasets, the largest change in performance is in the third convolution block (Table 4e), with the transition of information from the third to the fourth block of convolutions (Table 4f) only accounting for a small change in performance based on c. We use this to further demonstrate the merits of *Class Regularization* as a quantitative way of understanding the informative parts of the overall network, complementing the class-specific feature visualizations.

5. Conclusions

We have introduced *Class Regularization*, a method that focuses on class-specific features rather than treating each convolution kernel as class-agnostic. *Class Regularization* allows the network to strengthen or weaken layer activations based on the batch data. The method can be added to any layer or block of convolutions in pretrained models. It is lightweight, as the class weights from the prediction layer are shared throughout *Class Regularization* blocks. To avoid the vanishing gradient problem and the possibility of negatively influencing activations, the weights are normalized between a range given an *affection rate* value.

We evaluated the proposed method on three benchmark datasets: Kinetics-400, UCF-101, and HMDB-51; and three models: ResNet101, Wide ResNet50, and I3D. We consistently show improvement when using *Class Regularization*, with a performance gain of up to 2.45%. The achieved improvements were done with minimal additional computational cost over the original architectures. We also perform a statistical significance test to demonstrate that the outcomes of the different models are indeed based on the additional regularization, rather than being the result of incidental sampling.

Class Regularization can also aid in improving the explainability of 3D-CNNs. Qualitative visualizations reveal which spatiotemporal features are strongly correlated with specific classes. Such analyses can be made for specific layers and, as such, provide insight into the discriminative patterns that specific features represent.

Future works based on *Class Regularization* should aim towards addressing the feature variations of actions within the same classes, with a greater focus towards the temporal domain. The inclusion of global information and the calibration of local spatiotemporal patterns based on such information, as with *Squeeze and Recursion* blocks [26], shows a promising direction towards creating spatiotemporal features that can better treat the variations of human actions and interactions.

Author Contributions: A.S. was responsible for the methodology formalization. He has performed the software implementation and the experimentation. He is also the main contributor for the article. R.P. took part in the conceptualization of the work and edited the article. He also provided the funding for the work. R.C.V. supervised the work. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the Netherlands Organization for Scientific Research (NWO) with a TOP-C2 grant for "Automatic recognition of bodily interactions" (ARBITER).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Herath, S.; Harandi, M.; Porikli, F. Going deeper into action recognition: A survey. *Image Vis. Comput.* **2017**, *60*, 4–21. [CrossRef]
2. Stergiou, A.; Poppe, R. Analyzing human-human interactions: A survey. *Comput. Vis. Image Underst.* **2019**, *188*, 102799. [CrossRef]
3. Bau, D.; Zhu, J.Y.; Strobelt, H.; Zhou, B.; Tenenbaum, J.B.; Freeman, W.T.; Torralba, A. Visualizing and understanding generative adversarial networks. *arXiv* **2019**, arXiv:1901.09887.
4. Gilpin, L.H.; Bau, D.; Yuan, B.Z.; Bajwa, A.; Specter, M.; Kagal, L. Explaining explanations: An overview of interpretability of machine learning. In Proceedings of the International Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, 1–3 October 2018; pp. 80–89.
5. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
6. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
7. Simonyan, K.; Zisserman, A. Two-stream convolutional networks for action recognition in videos. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Montreal, QC, Canada, 8–13 December 2014; pp. 568–576.
8. Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; Van Gool, L. Temporal segment networks: Towards good practices for deep action recognition. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 8–16 October 2016; pp. 20–36.
9. Diba, A.; Sharma, V.; Van Gool, L. Deep temporal linear encoding networks. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2329–2338.
10. Wang, Y.; Song, J.; Wang, L.; Van Gool, L.; Hilliges, O. Two-Stream SR-CNNs for Action Recognition in Videos. In Proceedings of the British Machine Vision Conference (BMVC), York, UK, 19–22 September 2016.
11. Feichtenhofer, C.; Pinz, A.; Wildes, R. Spatiotemporal residual networks for video action recognition. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016; pp. 3468–3476.
12. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-Local Neural Networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
13. Ji, S.; Xu, W.; Yang, M.; Yu, K. 3D convolutional neural networks for human action recognition. *Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 221–231. [CrossRef]
14. Tran, K.N.; Gala, A.; Kakadiaris, I.A.; Shah, S.K. Activity analysis in crowded environments using social cues for group discovery and human interaction modeling. *Pattern Recognit. Lett.* **2014**, *44*, 49–57. [CrossRef]
15. Carreira, J.; Zisserman, A. Quo vadis, action recognition? A new model and the Kinetics dataset. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 4724–4733.
16. Hara, K.; Kataoka, H.; Satoh, Y. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 18–22.
17. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807.

18. Qiu, Z.; Yao, T.; Mei, T. Learning spatio-temporal representation with pseudo-3d residual networks. In Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 5534–5542.
19. Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; Paluri, M. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 6450–6459.
20. Lee, M.; Lee, S.; Son, S.; Park, G.; Kwak, N. Motion Feature Network: Fixed Motion Filter for Action Recognition. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
21. Tran, D.; Wang, H.; Torresani, L.; Feiszli, M. Video Classification With Channel-Separated Convolutional Networks. In Proceedings of the International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
22. Feichtenhofer, C.; Fan, H.; Malik, J.; He, K. SlowFast networks for video recognition. In Proceedings of the International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
23. Li, C.; Zhong, Q.; Xie, D.; Pu, S. Collaborative Spatiotemporal Feature Learning for Video Action Recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 7872–7881.
24. Stergiou, A.; Poppe, R. Spatio-Temporal FAST 3D Convolutions for Human Action Recognition. In Proceedings of the International Conference On Machine Learning and Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; pp. 183–190.
25. Lin, J.; Gan, C.; Han, S. TSM: Temporal Shift Module for efficient video understanding. In Proceedings of the International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October 2019–2 November 2019; pp. 7083–7093.
26. Stergiou, A.; Poppe, R. Learn to cycle: Time-consistent feature discovery for action recognition. *arXiv* **2020**, arXiv:2006.08247.
27. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015; pp. 448–456.
28. Stergiou, A.; Kapidis, G.; Kalliatakis, G.; Chrysoulas, C.; Poppe, R.; Veltkamp, R. Class Feature Pyramids for Video Explanation. In Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW), Seoul, Korea, 27–28 October 2019.
29. Stergiou, A.; Kapidis, G.; Kalliatakis, G.; Chrysoulas, C.; Veltkamp, R.; Poppe, R. Saliency Tubes: Visual Explanations for Spatio-Temporal Convolutions. In Proceedings of the International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019.
30. Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P. The Kinetics human action video dataset. *arXiv* **2017**, arXiv:1705.06950.
31. Soomro, K.; Zamir, A.R.; Shah, M. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv* **2012**, arXiv:1212.0402.
32. Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; Serre, T. HMDB: A large video database for human motion recognition. In Proceedings of the International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 2556–2563.
33. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
34. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. Imagenet: A large-scale hierarchical image database. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 248–255.

35. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. Large-scale video classification with convolutional neural networks. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014; pp. 1725–1732.
36. Chen, Y.; Kalantidis, Y.; Li, J.; Yan, S.; Feng, J. Multi-Fiber networks for Video Recognition. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.