Video hacked dataset for convolutionnal neural networks
Adrien Chan-Hon-Tong

To cite this version:
Adrien Chan-Hon-Tong. Video hacked dataset for convolutionnal neural networks. 2017. hal-01519630v2

HAL Id: hal-01519630
https://hal.archives-ouvertes.fr/hal-01519630v2
Submitted on 9 Jun 2017 (v2), last revised 5 Jul 2017 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract
Using successive images of a high framerate video to train convolutional neural networks can seem useless as successive images are so close that they contain almost the same information. It can be thought to be very close to the classical data augmentation technique consisting to add a small noise to the image.
However, we show in this paper that using successive images can provide significant increases of performances.
As annotating video can be done much more easily than annotating set of images, we argue that using successive images from video to train convolutional neural networks is at least that is something in context where data are missing or human annotation too expensive.

1 Introduction
1.1 Video hacked dataset
Perfect datasets should cover the variability of expected new data. Thus, perfect dataset should have both large size and large variance. However, manually annotating large datasets require large human time.
Crow engineering tries to solve this problem using large number of human but this is not always possible (for example if data are hard to share or should not be diffused).
Semi-automatic annotation tries to increase human productivity. But, there is few chance to provide a real productivity gap if data have large variances as tools designed on a piece of the dataset may not perform well on the other part.
Thus, there is an interest for algorithms which could be able to learn from less data, or (this is the purpose of this paper), from correlated data i.e. large size but low variance dataset.
Indeed, in image context, such dataset can be easily created by using video trick: using temporal consistency on high frame rate video allows to propagate freely expensive human annotation from one image to the
(a) a remote sensing image  (b) a semantic mask

Figure 1: Example of semantic segmentation of an image. Each color in the semantic mask is expected to correspond to a kind of object in the image. For example, boats in the image are pink in the semantic mask while cars are yellow.

next while increasing quickly the size of the dataset. Such, video dataset can be done to use deep learning pipelines working directly on the video. But this is not the purpose of this paper. Here, we focus on whether using such video hacked dataset is relevant for convolutionnal neural networks (CNN) working on image (and not video).

1.2 Selected use case

More precisely, we focus on the relevancy of such video hacked datasets to learn CNN for binary semantic segmentation.

Semantic segmentation (see 1) is the goal of producing semantic mask of an image. Or, in other words, the goal of deciding a semantic label for each pixel of an image. In our experiment, we will perform such segmentation task but only with two classes. Again, the pipeline is designed to process images: each mask is completely estimated using the current image only. In this context, we focus on the relevancy of using video hacked datasets to increase performances of CNN.

Very briefly, a little explanation of why we focus on binary semantic segmentation: our goal is to build on the top of a segment before detect approach (e.g. [1]). Object detection in image is a well studied computer vision problem (see figure 2). Recently, deep learning techniques [6, 14] have boosted object detection performances. [6] use a region proposal to transfer deep learning classification technique to object detection, while [14] tries to directly predict bounding boxes. However, both box proposal and direct prediction of boxes have to deal with unstructured output: the number of boxes is data dependant and values in box encoding have different semantic (offset or size). Optimizing deep learning is much
straightforward in semantic segmentation: output is a highly structured
tensor with predefined size and a number of channels corresponding to
the number of types of objects, which can be easily matched against the
manual annotation.

However, we think that the question tackled by this paper is not re-
stricted to binary semantic segmentation: this paper is about using suc-
cessive images of a video for many CNN tasks like classification, detection,
segmentation...

1.3 Design of the experiments

The contribution of this paper is to present an experimental protocol to
evaluate the relevancy of video hacked datasets.

For this purpose, we select (or create) several datasets consisting of
very different annotated videos. Then, we perform several kinds of exper-
iments to measure the performance of fixed CNN trained with different
training datasets extracted from the video, e.g. we measure the perfor-
mance of segmentation pipelines while increasing the number of successive
images used from each video. This example of experiment should have a
clear output: no performance augmentation while increasing the number
of images would argue that using video hacked dataset is not relevant.

However, there is a lot of possible pitfalls, and, our main concern in
the following is to try to avoid these.

The first one is that if the performance is too low independently of
the number of images, one can wonder if the problem is no just too hard
independently from the setting. Same ambiguity appears if it is too high.

Then, here we aim to compare a common learning pipeline trained with different data (and tested on common ones). This raises an issue about at what point two runs can be considered as implementing a common pipeline. When the learning algorithm is kind of global/convex, comparison of is less ambiguous. But, global/convex learning pipelines would have to be very scalable to process datasets whose sizes vary from 1 to 400. And, degrading the training parameter (to increase speed) as the size augment may lead to situations where we do not know if there is no performance increase because additional images are not interesting or because we degrade the training parameter.

Inversely, when training with tools like stochastic gradient descent, we first lost reproducibility, and, we may introduce tuning bias: solver parameters will (more of less consciously) be tuned on the smallest dataset and may artificially bias performances.

To strengthen the output of our experiment, we chose to use a lot of different datasets and state of the art deep learning to handle ambiguity concerning the relevancy of the datasets to answer the question. Then, we chose to perform both convex based experiment (pre trained deep learning being used like feature extractor) and fine-tuning experiment (straightforward deep learning training) to handle ambiguity concerning tuning bias in training.

To keep sufficiently low total training time, not all experiment will be realized with all modality, but we will try to maintain a global consistency in the experiments.

The structure of this paper is the following. In next section, we describe related works. Then, sections 3 and 4 describe experiments about the performance of segmentation, section 3 for convex based experiments and section 4 for fine-tuning ones, before the conclusion and perspectives in section 5.

Finally, this work has been done by a team all author will be listed in final version.

2 Related works

2.1 Semantic segmentation

Semantic segmentation is a growing paradigm [2, 15]. Ones of the most important datasets of semantic segmentation are designed for autonomous car issues: CITYSCAPE [3] and SYNTHIA [16].

For aerial images, most works related to convolutional neural network CNN for semantic labelling use 3 channels networks designed for RGB, fine tuned from a model trained on the IMAGENET dataset [13, 10, 9].

However, the situation is very different between autonomous car contexts with very large datasets (e.g. CITYSCAPE and SYNTHIA), and, aerial contexts especially for small object. For example, the 2015 IEEE data fusion contest contains less than 400 car instances which limits the stability of algorithm evaluation (e.g. [7]).
Even if there is a complete field of research to use unsupervised algorithms to take advantage of the very large available data (Sentinel2 huge data for example can be freely downloaded), we could bet that performance will not largely increase until sufficient coherent and annotated dataset will be published.

But, annotating large annotated datasets is a very expensive human process. For applications which may generate huge economic feedback like autonomous car, large consortium can be formed to pay the annotation.

2.2 Semi automatic annotation

One possible cheap answer to this lack of annotation is semi automatic annotation where human annotator is helped by computer vision algorithm.

It includes active learning where computer vision proposes best next thing to annotate (e.g. [21]). It includes crow engineering [18, 20]. It also includes optimization of human time [17].

However, crow engineering is not possible for secret data (like military images) or when annotating requires expert knowledge. Optimization of the human time and active learning are relevant when sufficiently good algorithms are already available. In [17], the idea is just to move from a sparse annotation to a dense annotation by using deep network learnt on the same kind of datasets.

Also, as learning algorithm are more and more complex, it is not trivial to perform an efficient online interactive semi automatic annotation system. To our knowledge, there is no widely used system whose efficiency has been successfully evaluated.

One other possible cheap answer is the use of temporal consistency when annotating a video. The interest of video is that information can be propagated (by tracking [22] or optical flow [19]) from one frame to the next allowing to save costly human annotation when the propagation behaves correctly. This propagation only uses low level clue, which would not be sufficient if not helped by the temporal consistency.

For the purpose of experiments (including this), we develop a tracking based annotation tool. This tool is very close to [18] but use off the shelf computer vision tracker (opencv implementation of dsst tracker [4]) to propagation bounding boxes from one frame to the other. Human correction is only needed when tracker fails to be sufficiently accurate. This happens about every 20 images.

This last answer is sufficiently mature to help annotation. However, it raises the question focused by this paper: does this video trick increase the performance or just the dataset size ? Or in other words, is it relevant to learn from highly correlated data ?

Does video hacked dataset perform better than classical add random Gaussian noise data augmentation (we do not review data augmentation technique as we implement this noise only) ?

2.3 Scalability issues in SVM

As explained in section 1, to strengthen the conclusion of this experiments, we perform both deep network training (more precisely finetuning) and
convex training. In this convex training, deep networks are used with pre trained weight as feature extractors. We acknowledge this is usually considered as worse than finetuning [6] but here we are interested by the relative performances while increasing the number of images and not by the absolute performances.

The interest of convex training is strong reproducibility and possibility to use off the shelf parameters.

However, as strange as it can seem, deep learning is much more scalable than convex learning, despite its enormous computational needs. Even using liblinear [5] which is recognized as a one of the fastest and most reliable svm implementation, we quickly hit the solver limit.

Currently, forwarding a few VGA images into a segnet like network [2] and saving the resulting features mask in liblinear format yet leads to a very large file. Worse, processing these files with liblinear requires too large RAM resource (even server cannot meet such requirement). In order to perform our experiment, we have no choice but to rely on liblinear-block [23] or liblinear + average (like in [23]).

We acknowledge that, in [23] experiments, liblinear-block is used to process datasets 20 times larger than memory, whereas in our experiments, the expected dataset size is close to 400 times the memory available in a standard computer. However, we do not find better implementation (which meets both our scalability and reproducibility requirements).

3 SVM based CNN adaptation on video hacked datasets

3.1 Global pipeline

In this section, we use a pre trained CNN to extract feature map for each training image. Then, we train SVM for semantic segmentation (see figure 3).

We rely on vgg16 in a fcn fashion: we forward each image into vgg16 (pretrained on imagenet), extract several layers (conv12, conv22, conv33, conv43 and pool5). The network is described in figure 4. We resize all extracted layers to the ground truth size, obtaining a feature map in which each pixel is described per a vector. We learn a svm with each pixel of the feature map being a training point.

We allow the ground truth to be eventually smaller than the original image when spatial accuracy of the human annotation is not relevant enough.

We learn a svm with liblinear or liblinear block when we reach the RAM limit (8Go).

We also learn a svm with a simple stochastic gradient descent (sgd): we loss reproducibility but it is much more faster and useful for some of the experiments.
Feature map spatial size is not related to network structure: it is resized to the ground truth.

If we have $n$ training images then the svm learn from $n \times w' \times h'$ points: e.g., for $n = 100$, $w' = 320$ and $h = 240$ there are yet 7680000 points for the svm.

3.2 Datasets

There is a lot of semantic segmentation datasets. But most of them (MSCOCO [8], IEEE data fusion contest [7], ...) are images datasets. They are thus irrelevant to focus on video hacked datasets.

One of the most important video datasets is CITYSCAPE. It is composed of 25000 HD frames pixelwise annotated into 30 semantic classes. Images are taken from a car. The Dataset is clearly designed for autonomous car research. However, frame rate is too low to evaluate the ability to learn from simultaneous images: typically no pixel from an image are present in the following. We will evaluate our pipeline on this dataset for comparison with the state of the art only.

SYNTHIA is a dataset very close to CITYSCAPE (except that images are synthetic). However, SYNTHIA videos do not respect temporal consistency. There are thus completely unusable for our experiment.

Thus, to tackle this paper issues, we rely on MOT 2016 (MOT16) [11], VIRAT aerial dataset [12] (not the video in video surveillance setting. We use only a part of the aerial videos. We annotate it ourselves using vidannot - annotations are planned to be released) and a private dataset (that may or may not be released) which is very close to VIRAT but with HD image instead of low resolution images.

MOT16 (https://motchallenge.net) is a multi objects tracking dataset: detection are provided, the goal is to keep temporal id on the detection. Here, we use it only to produce a semantic mask corresponding to the detections (we only keep person detection). We convert the detection ground truth into a semantic segmentation ground truth by considering that all pixels in a detection are from class 1 and all pixels outside detections are from class -1. The training part of MOT16 is composed of 6 HD
The only layers with weight are the convolutions which are initialized with vgg-on- imagenet weights. Each $W \times H \times 3$ image leads to several layers whose dimensions are typically $W \times H \times K_1$, $\frac{W}{2} \times \frac{H}{2} \times K_2$, $\frac{W}{4} \times \frac{H}{4} \times K_3$, $\frac{W}{8} \times \frac{H}{8} \times K_4$. All these blobs are resized to $W \times H$ and concatenated, leading to a $W \times H \times (K_1 + K_2 + K_3 + K_4)$ blob. Each pixel is classified independently using the corresponding $K_1 + K_2 + K_3 + K_4$ values leading to a $W \times H$ mask.
videos taken from a pedestrian or car or surveillance camera (see figure 5).

For MOT16, we will use accuracy, gscore and iou score to evaluate all algorithms. Accuracy is the stablest measure, gscore is the product of precision per recall and iou score is the same as for the CITYSCAPE LEADERBOARD (this is not related to the IoU of 2 boxes in detection to know if a matching is possible, this is a score computable from the confusion matrix that aim to give the same weight on positif and negatif while acknowledging both two types of error).

The VIRAT aerial dataset is a set of videos which are low resolution and contain camera motions, highly textured background and very small object. As no public annotation has been released for this dataset, we annotated a subset of the frames in a person detection setting. We convert the ground truth in the same way than for MOT16. In order to provide a diversity of situations, we chose to annotate about thirty sub videos of 400 frames containing at least one person distributed over the dataset (but discarded infrared images). Figure 6 shows examples of images from this sub dataset.

For VIRAT, we will use only gscore and IoU score (like in CITYSCAPE LEADERBOARD) to evaluate all algorithms. Accuracy is not relevant as 99.2% of the pixel are background pixel.

3.3 Results

3.3.1 Sanity-check

First, in order to ensure that we use a sufficiently state of the art pipeline, we evaluate our pipeline on a small subset of CITYSCAPE which is a state of the art dataset.

This evaluation is not done to compare with CITYSCAPE result but only to show that results are on the same order of magnitude.

Our network is very close to [2] which reaches around 80% of IoU category. Here, we make the problem binary by merging all object categories vs surface ones. We only use either 1 or 20 images per training video and we test on 60 images extracted from val video. Our liblinear pipeline reach 87% of IoU meta category (83% for our SGD pipeline). This result is not directly comparable with the ones of the cityscape leader board for three reasons. It is measured on a subset of val data and not on test data, then we measure IoU on the two meta categories and not on class
Figure 6: Illustration of the subset of VIRAT (plus annotation).
pipeline | accuracy | iou  | gsore  
---|---|---|---
liblinear 1 | 90 (96) | 33 (76) | 29 (76) 
liblinear 20 | 91 (95) | 36 (67) | 31 (65) 
sgd 1 | 88 (92) | 12 (42) | 10 (39) 
sgd 20 | 89 (92) | (38) 53 | 30 (49) 

Table 1: Results 1 vs 20 of svm on MOT16.

score are given in percentage. Number between bracket correspond to training score which are relevant in this paper.

or category (it would be lower on category), but, we only use 12 training images instead of 25000 (so it would be higher if we perform the complete training with fine tuning).

This evaluation is just here to shows that our pipelines is sufficiently close to the state of the art to ensure validity of the following experiments.

3.3.2 Different types of experiments

We perform 3 kind of experiments to evaluate the impact of video hacked datasets

- we compare the performance of a cnn trained with 1 image per video with the same cnn trained with 20 successive images per video
- we also compare the performance of a cnn trained with the images \{0, 20, 40, ..., 380\} of each videos with the same cnn trained with 400 successive images per video (notice that coupled with the first experiment this provide also an information about the performance of 1 image per video to 400 image per video)
- we finally compare the performance of a cnn trained with 20 successive images per video with the same cnn trained with 20 randomly noised images computed from the image 1 (on the 20) - so the number of image is the same but in one hand it is real successive images and on the other hand randomly generated one. More precisely, we did this experiment both for image 1 to 20 vs 20 images generated from image 1 and for images 1 to 400 versus 400 images generated from images \{0, 20, 40, ..., 380\}

Figure 7 gives an overview of this 3 types of experiments.

3.3.3 performances vs frames per video

In all experiment, the name of the pipeline is the solve (liblinear or sgd) plus the number of images used (1, 20, 1by20, 400). We use different notations 20 and 1by20 to make a distinction between using image 1 to 20 and image 1, 21, 41, ..., 381.

Performance of all experiments are reported in tables 1, 2, 3 and 4. See section 3.3.2 for the description of the two experiments 1 vs 20 and 20 vs 400.
(a) 1 vs 20 experiment: we evaluate the performance of a cnn trained with image 1 from each video versus image 1 to 20 from each video.

(b) 20 vs 400 experiment: we evaluate the performance of a cnn trained with image 1, 21, 41, ..., 381 from each video versus image 1 to 400 from each video.

(c) successive images vs noised images: we evaluate the performance of a cnn trained with image 1 to 20 from each video versus 20 images randomly noised from image 1 from each video.

Figure 7: Overview of the 3 types of experiments performed.
<table>
<thead>
<tr>
<th>pipeline</th>
<th>accuracy</th>
<th>iou</th>
<th>gscore</th>
</tr>
</thead>
<tbody>
<tr>
<td>liblinear 1by20</td>
<td>87 (90)</td>
<td>40 (52)</td>
<td>34 (48)</td>
</tr>
<tr>
<td>liblinear 400</td>
<td>89 (88)</td>
<td>45 (49)</td>
<td>39 (43)</td>
</tr>
<tr>
<td>sgd 1by20</td>
<td>87 (91)</td>
<td>35 (51)</td>
<td>27 (45)</td>
</tr>
<tr>
<td>sgd 400</td>
<td>89 (92)</td>
<td>48 (49)</td>
<td>30 (49)</td>
</tr>
</tbody>
</table>

Table 2: Results 20 vs 400 of svm on MOT16.

score are given in percentage. Number between bracket correspond to training score which are relevant in this paper.

<table>
<thead>
<tr>
<th>pipeline</th>
<th>iou</th>
<th>gscore</th>
</tr>
</thead>
<tbody>
<tr>
<td>liblinear 1</td>
<td>17 (50)</td>
<td>8 (48)</td>
</tr>
<tr>
<td>liblinear 20</td>
<td>12 (34)</td>
<td>6 (26)</td>
</tr>
<tr>
<td>sgd 1</td>
<td>13 (39)</td>
<td>9 (32)</td>
</tr>
<tr>
<td>sgd 20</td>
<td>11 (37)</td>
<td>6 (30)</td>
</tr>
</tbody>
</table>

Table 3: Results 1 vs 20 of svm on VIRAT.

score are given in percentage. Number between bracket correspond to training score which are relevant in this paper.

<table>
<thead>
<tr>
<th>pipeline</th>
<th>iou</th>
<th>gscore</th>
</tr>
</thead>
<tbody>
<tr>
<td>liblinear 1by20</td>
<td>12 (20)</td>
<td>4 (9)</td>
</tr>
<tr>
<td>liblinear 400</td>
<td>20 (21)</td>
<td>4 (11)</td>
</tr>
<tr>
<td>sgd 1by20</td>
<td>21 (26)</td>
<td>13 (17)</td>
</tr>
<tr>
<td>sgd 400</td>
<td>20 (31)</td>
<td>12 (11)</td>
</tr>
</tbody>
</table>

Table 4: Results 20 vs 400 of svm on VIRAT.

score are given in percentage. Number between bracket correspond to training score which are relevant in this paper.
These results are interesting especially on MOT because performances are in the correct range (not too low, not too high). Thus, we can state that using video hacked dataset significantly (even if not largely) increases performances from 1 to 20 and from 1 by 20 to 400.

Performances on VIRAT are low especially the gscore which is strongly sensible to the trade off between precision and recall when performance are low. In this condition, it is harder to conclude to the relevancy of adding successive frames. However, iou score still slightly increases from 1 to 400 even if not linearly.

In our opinion, these experiments argues that video hacked dataset can increase performance of deep learning binary segmentation pipeline.

3.4 Successive images versus data augmentation

The global idea is to have in one side 20 successive images and on the other side 20 randomly noised images generated images. More precisely, with add a gaussian noise with same variance that in the set of successive images. Images generated are converted into 8 bits images to enter the CNN.

As we want to make multiple runs of the training to average random noise effect, we perform this type of experiment with sgd only (and not with liblinear). The great advantage is that the image can be generated on the fly with SGD.

The results of these experiment is very clear: successive image are largely better than random ones on all the evaluated setting. In our experiment, using these gaussian noised images often even decrease the performance of using only the raw image.

The question of either an other types of noise (or a noise not converted into 8 bits) could have been better is out of the scope of this paper. But, this result show, at least, that contrarily to what one could have though, successive image of a video are much more useful than simple data augmented images.

4 CNN on video hacked datasets

4.1 Global pipeline

In this section, we did a second time some of the experiments described in section 3 but in a straightforward deep learning fashion (see figure 8). This training fashion is expected to be more efficient than svm adaptation both in performance and time (but much less reproducible and robust)

Network are designed to directly produce an output with same shape as the ground truth (we work on random crop of size 256x256). Backpropagation is directly performed by computing the gradient corresponding to the loss between the ground truth and the network output.
Figure 8: Global pipeline for finetuning experiments.
Network is trained on random crop extracted from the training images (size of crops is 256x256 if not detailed). Network directly outputs a layer with same shape than ground truth. The loss is the average on all the pixels of the cross entropy. Learning is done by stochastic gradient descent on a standard forward backward scheme. Training is done with nvidia DIGITS tools.

4.2 Networks
We evaluated different state of the art networks: fcn like (the same as the svm experiment), unet [15], and a deeper unet (we add a level to the unet structure).
See original publications for the details about these networks, we reproduce figures from [15] only for the sake of reading in figure 9.
All tested networks are based on vgg and are finetuned from vgg initialisation.

4.3 Results
Finetuning experiments are not slower than svm one (in fact nearly always faster) but these experiments take the handle on expensive (and thus shared) GPU hardware. So, it was not easy for us to replicate all the slow (but CPU based) svm experiments. Notice that one may need more than 8Go of GPU memory to reproduce our experiments (typically a NVIDIA Titan P).
So, we only evaluate the increase of performances while increasing the number of successive images between 1, 20 and 400.
Available results are presented in tables 5 for MOT16 and 6 for VIRAT.
First, these results are surprising because finetuning is expected to be much better than feature extraction + svm (which corresponds to a finetuning of the last layer only). But, this is not that clear on our datasets: it may be due to the crop effect (full image can not be handled so crop of size 256 are extracted on a regular grid). Or it may be due to the difficulty of our dataset especially VIRAT.
This figure is purely copied from the original unet paper. See the original paper for details of the structure.

<table>
<thead>
<tr>
<th>pipeline</th>
<th>precision</th>
<th>recall</th>
<th>gscore</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcn 1</td>
<td>25 (95)</td>
<td>12 (89)</td>
<td>3 (85)</td>
</tr>
<tr>
<td>fcn 20</td>
<td>33 (99)</td>
<td>18 (99)</td>
<td>6 (99)</td>
</tr>
<tr>
<td>unet 1</td>
<td>67 (41)</td>
<td>0 (02)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>unet 20</td>
<td>59 (99)</td>
<td>14 (99)</td>
<td>8 (99)</td>
</tr>
<tr>
<td>unet 400</td>
<td>19 (96)</td>
<td>01 (43)</td>
<td>0 (41)</td>
</tr>
<tr>
<td>deep unet 1</td>
<td>100 (100)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>deep unet 20</td>
<td>69 (100)</td>
<td>23 (100)</td>
<td>16 (100)</td>
</tr>
<tr>
<td>deep unet 400</td>
<td>89 (98)</td>
<td>33 (89)</td>
<td>30 (97)</td>
</tr>
</tbody>
</table>

Table 5: Finetuning results on MOT16.

<table>
<thead>
<tr>
<th>pipeline</th>
<th>precision</th>
<th>recall</th>
<th>gscore</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcn 1</td>
<td>0 (91)</td>
<td>0 (66)</td>
<td>0 (60)</td>
</tr>
<tr>
<td>fcn 20</td>
<td>0 (27)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>unet 1</td>
<td>70 (99)</td>
<td>11 (99)</td>
<td>9 (99)</td>
</tr>
<tr>
<td>unet 400</td>
<td>23 (70)</td>
<td>91 (72)</td>
<td>21 (50)</td>
</tr>
</tbody>
</table>

Table 6: Finetuning results on VIRAT.
However, the most important point, in our opinion, is that these experiments confirm the trend observed in svm ones: performance tends to increase when increasing the number of successive images used from the videos. This is especially clear for unet 1 to unet 400 on VIRAT with a gscore who jumps from 9% to 21%. This is also especially clear for deep unet on MOT who reaches 0%, 16%, 30% when using 1, 20, 400 images.

5 Conclusion

This paper focuses on video hacked datasets for image semantic segmentation. The question is about the relevancy of using video on which temporal information can be used to help human annotation to train purely image semantic segmentation pipeline (especially deep learning pipelines).

In our experiments, using this video trick increases performance of the pipelines whereas adding gaussian noise (to form a dataset with the exact same size) does not. Of course, using uncorrelated images will be more efficient but using video may still be useful if data are naturally collected this way and allows to save annotation time.

As we do not talk about this point before, we have to acknowledge that this small performance augmentation comes at a price of a very large computation augmentation (typically 1 to 400), but this consideration is out of the scope of this paper where we care only about the cost of human annotation.

This paper should be continued to investigate if learning from video hacked dataset provides some specific features, we think for example about robustness to aliasing (which has currently not been observed but is still a good candidate to explain the increase of the performances).

References


