Reproduction of *Privacy-Preserving Action Recognition via Motion Difference Quantization*

Ciprian Bîcă Tadas Žaliauskas
Yizhen Zang Žygis Liutkus

June 2025

Notes

Incorporate the following into corresponding sections.

- Paper doesn't provide an answer to which 3D ResNet architecture is used - i3d, slowfast, or any other. After looking into the author's code, it becomes obvious that it was i3d. Not sufficiently detailed if any transformations are being applied during training.
- Unclear which layers are frozen in pre-trained ResNet in order to fine-tune it to another dataset (the last one?)
- Code is not structured in layers, it's hard to find where classes responsible for Blur, Quantization, and Difference are located
- No details provided about dataset splits for training/validation/testing
- How training is done: there are 2 steps (1) P is frozen, E and T are trained together; 2) E and T are frozen, P is unfreezed and trained) with different components being frozen and for each of those steps, different criteria are applied for each batch?
- Do values need to be normalized at 2 places: in ResNet as well or only initially?
- Validation - need to fine-tune ResNet after each epoch of train?
- Scalar hardness term, what's x and y in blur?
- Naming differences - model_budget, model_degrad in paper provided code. Different objectives than in paper (fix encoder, and train target and privacy net? Inaccurate loss function loss_target-2*entropy_budget+10.). Budget stands for P from the paper. Not clear if for privacy ResNet, random frame is taken or logits are averaged across all frames. What was mean and std used for normalization. No training acc./loss curve, unclear what's expected progress when reproducing. Not detailed hardware setup. Incorrect Gaussian function in code (multiplier).

1 Introduction

The increasing use of computer vision systems in daily life has raised privacy concerns [1]. Although these systems are typically designed to recognize and respond to human activities, they also capture sensitive information, such as one's identity and gender [2]. This leads to a key challenge: how can we preserve privacy while still allowing the systems to perform their task of recognizing human actions?

To address this, Kumawat and Nagahara proposed the BDQ encoder, a lightweight and privacy-preserving framework for action recognition [3]. The encoder consists of three sequential modules, Blur, Difference, and Quantization. It's designed to suppress sensitive visual cues while keeping the motion patterns needed for action recognition. The model is trained using adversarial learning to balance the two goals, maximizing action recognition accuracy and minimizing privacy prediction accuracy.

In this project, we aim to reproduce the BDQ framework and evaluate it on an additional dataset. Our work consists of four main components:

- We re-implement the BDQ encoder from scratch, following the architecture and training strategy described in the paper. Our implementation is available at `https://github.com/cactusutcac/frmdl25-6`, with clear module and function names, well-structured configuration files, and a comprehensive README to facilitate reproducibility.
- We evaluate our implementation of the KTH dataset [4] to validate whether similar performance can be achieved.
- We test the generalization of BDQ on a new dataset, IXAMS [5], which includes multiple viewpoints.
- We conduct an ablation study on our BDQ implementation to test whether the privacy attribute prediction module in the adversarial setup is effective. Specifically, we examine whether the BDQ encoder actually reduces the ability of a separate model to recover sensitive identity information.

By reproducing the original setup and extending it to new settings, we aim to understand the strengths and possible limitations of the BDQ encoder and its potential for broader applications in privacy-preserving action recognition.

2 Framework Reproduction

Figure 1 illustrates the architecture of the original BDQ framework, which forms the basis of our reproduction. The BDQ encoder E, composed of Blur, Difference, and Quantization modules, processes input video frames to suppress privacy-revealing cues while preserving motion dynamics critical for action recognition. The adversarial training scheme jointly optimizes E, the action recognition model T, and the privacy attribute predictor P. During the training, E is encouraged to encode features that are discriminative for actions but uninformative for identity. In this section, we detail our implementation of each component and describe how we adapted the training and validation procedures to align with the original setup.

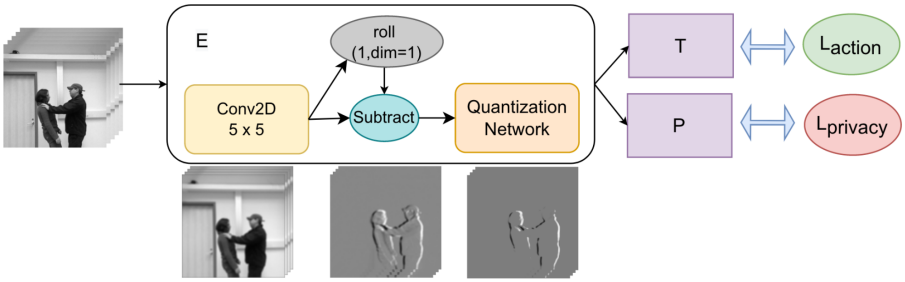


Figure 1: The BDQ encoder architecture and adversarial training framework for privacy-preserving action recognition. Figure reproduced from [3].

2.1 BDQ Encoder (E)

Following the original paper, we implemented the three sequential modules of the encoder. The encoder outputs a privacy-filtered sequence of tensors that retain the motion patterns necessary for action recognition while reducing the risk of identity leakage.

Blur The Blur module applies a spatial Gaussian blur to each frame using a 5×5 kernel. The standard deviation σ is initialized to 1.0 and is implemented as a learnable parameter. This allows the encoder to adapt the strength of blurring during training, balancing the suppression of identity-revealing details with the preservation of action-relevant structure. Blurring is performed via depthwise separable 2D convolution using PyTorch's (`F.conv2d`) for efficiency, and a 1D Gaussian kernel is computed and applied in both horizontal and vertical directions. This step ensures that sensitive identity cues at edges are smoothed out before motion is computed.

Difference The Difference module computes pixel-wise intensity differences between consecutive frames in the temporally ordered video sequence. Given a tensor of shape $[B, T, C, H, W]$, it returns $[B, T-1, C, H, W]$ where each output frame is the difference between two consecutive blurred frames. This implementation does not involve learnable parameters and operates purely through tensor slicing and subtraction. This operation enhances motion information while removing static content, making it useful for capturing temporal dynamics without relying on computationally expensive optical flow.

Quantization The Quantization module further processes the motion difference frames using a differentiable soft quantization function. It approximates a Heaviside step function with a sigmoid, applied across 15 learnable bin centers. Each pixel value is mapped to a soft histogram across these bins: $y = \sum_{n=1}^{N-1} \sigma(H(x - b_i))$, where H is a scalar hardness term (default is 5), and $\{b_i\}_{i=1}^{14}$ are the learnable bin centers. These centers are initialized between 0.5 and 14.5 across 15 bins. Inputs are normalized to $[0, 15]$. This operation compresses motion representation while suppressing residual low-level cues. The replacement with the sigmoid $\sigma()$ function makes it fully differentiable for backpropagation. This facilitates adversarial training where the encoder is jointly optimized to retain action-relevant information while suppressing identity cues.

2.2 Action Recognition Model (T)

To perform action classification from BDQ-encoded video inputs, we used a 3D convolutional neural network based on the I3D-ResNet50 architecture from PyTorchVideo. This module is responsible for predicting action classes from spatio-temporal features retained after privacy-preserving transformations.

Architecture The model is instantiated using the `i3d_r50` backbone pre-trained on the Kinetics-400 dataset as done by the original paper. During initialization, the model is set to evaluation mode and optionally fine-tuned by replacing only the final classification head.

Preprocessing Prior to classification, each input video undergoes a series of transformations to align with the network's input expectations.

- *Temporal Subsampling*: uniform sampling of 8 frames from each video.
- *Normalization*: per-channel normalization using mean of 0.45 and standard deviation of 0.225.
- *Spatial Scaling*: resize shortest side to 256 pixels.
- *Center Crop*: extract a 256×256 central patch.

2.3 Privacy Attribute Prediction Model (P)

Following the original paper, we used a 2D convolutional model P to predict sensitive attributes such as identity. This component acts as an adversary in the training process and serves as an evaluation probe during validation.

Architecture The model P is instantiated as a ResNet-50 network from the `torchvision` library and is initialized with pre-trained weights on ImageNet. All layers except the final classification head are frozen to retain the generic feature extraction capabilities. Only the final fully connected layer is trainable and adapted to the number of privacy classes.

Preprocessing The model accepts BDQ-encoded frames of shape $[B, C, H, W]$ and these inputs are normalized with ImageNet statistics to match the ResNet training distribution. For multi-frame video clips, each frame is processed independently by the ResNet backbone. The per-frame softmax outputs are then averaged to obtain a single prediction per clip. This frame-wise average assumes identity information is consistently embedded across frames while action-specific variance is temporally dynamic.

2.4 Adversarial Training

todo

2.5 Validation

todo

3 Datasets

We evaluate our BDQ encoder on two established datasets for human action recognition, the KTH dataset (used in the original paper) [4] and the IXMAS dataset [5].

3.1 KTH Dataset

3.1.1 KTH Dataset Overview

The KTH human action dataset, introduced by Schuldt et al. [4], is one of the earliest benchmarks for video-based action recognition. It contains 2391 video sequences of six types of actions: walking, jogging, running, boxing, hand waving, and hand clapping. These actions are performed by 25 different subjects across four scenarios: outdoors, outdoors with scale variation, outdoors with clothing variation, and indoors.

Each video is recorded using a static camera at 25 frames per second, with an average duration of four seconds. The sequences have a relatively low spatial resolution (160×120 pixels) and clean, homogeneous backgrounds. Although the dataset lacks viewpoint diversity and camera motion, it remains widely used due to its simplicity, consistent annotations, and the availability of multiple performance baselines.

3.1.2 KTH Dataset Preprocessing

To prepare the KTH dataset for training the BDQ encoder, we implemented a custom preprocessing pipeline that extracts action clips, normalizes spatial and temporal dimensions, and applies consistent transformations. As the original BDQ paper omits details on how the KTH dataset was handled, and the available public code is ambiguous on preprocessing, we designed a clear and reproducible pipeline based on metadata provided in the `00sequences.txt` file.

Clip Extraction Each original video in the KTH dataset contains 4 action instances annotated by start-end frame indices in `00sequences.txt` (available in Recognition of human actions. We parsed this metadata using a dedicated script, extracted individual clips, and converted them into a structured JSON file. Each clip entry includes the action label, subject ID, scenarios, and the frame range for the clip. It's never mentioned in the original paper or code how the KTH dataset was applied, so we decided to use the second, third, and fourth clips of each video for training, validation, and testing respectively.

Video Decoding and Frame Extraction Given the start and end frames of each clip, we used OpenCV to decode the corresponding segment from the `.avi` files. Each frame is converted to RGB using PIL and then to PyTorch tensors. This is handled by a custom PyTorch `Dataset` class `KTHBDQDataset`.

Transformations and Normalization To prepare the clips as input to the BDQ encoder, we apply the following transformations.

- *Temporal Center Cropping*: selects a fixed number of sequential frames (e.g., 32) from the center of each clip to maintain motion continuity.
- *Multi-Scale Spatial Cropping*: applies spatial crops at multiple scales using a fixed set of offsets.

- *Pixel Value Rescaling*: scales raw pixel values to the range $[0, 1]$ to prevent saturation and help network stability.
- *Channel-Wise Standardization*: standardizes pixel values using dataset-specific mean and standard deviation values precomputed from the original BDQ implementation.

Output Format The final output of the preprocessing pipeline is a PyTorch tensor of shape $[T, C, H, W]$, where T is the number of frames, $C = 3$ is the color channel count, and H, W are the height and width after cropping. Each clip is returned alongside its action label and subject ID.

3.2 IXAMS Dataset (New)

3.2.1 IXMAS Dataset Overview

The INRIA Xmas Motion Acquisition Sequences (IXMAS) dataset, introduced by Weinland et al. [5], provides a more challenging setting by incorporating multi-view recordings and unconstrained action execution styles. It consists of twelve actions, such as check watch and cross arms, each performed three times by ten actors (five male and five female). All actions were captured simultaneously from five calibrated, static cameras placed at different viewpoints to enable multi-view action recognition.

Unlike KTH, IXMAS actors were given minimal guidance on how to perform the actions, leading to natural variations in execution style and orientation. The dataset includes both intra-class and viewpoint variability, making it well-suited for evaluating privacy-preserving methods that decouple identity from motion.

To better align with the experimental setup of the original BDQ paper, we manually selected six action classes (check watch, point, kick, sit down, get up, and walk) and kept only two viewpoints per subject, i.e., those offering the most frontal perspectives, for each selected action. We did this to strike a balance between computational feasibility and sufficient coverage of inter-subject and inter-view variations.

3.2.2 IXMAS Dataset Preprocessing

To adapt the IXMAS dataset for training the BDQ encoder, we implemented a custom pipeline tailored to its multi-camera and free-form nature.

Clip Parsing and Metadata Construction Given the absence of official annotations for frame boundaries, we treated each video file as a complete clip. Our selection strategy includes only six of the twelve original action classes and two views per subject-action pair to reduce computational overhead. Each video filename in IXMAS encodes metadata such as subject identity, action class, repetition index, and camera viewpoint. We extract this metadata with a parser and organize it into structured clip entries. A total of six clips are expected per action per subject (three repetitions with two selected viewpoints each). Each entry records the full frame range from the original video and assigns a deterministic split: the first three clips are used for training, the fourth for validation, and the remaining two for testing. These entries are stored in `ixmas_clips.json` for use during training.

Subject Indexing To standardize identity labels across diverse subject naming conventions (e.g., `alba1`, `daniel3`), we mapped each subject name to a unique numeric index.

The other steps are the same as detailed in Section 3.1.2.

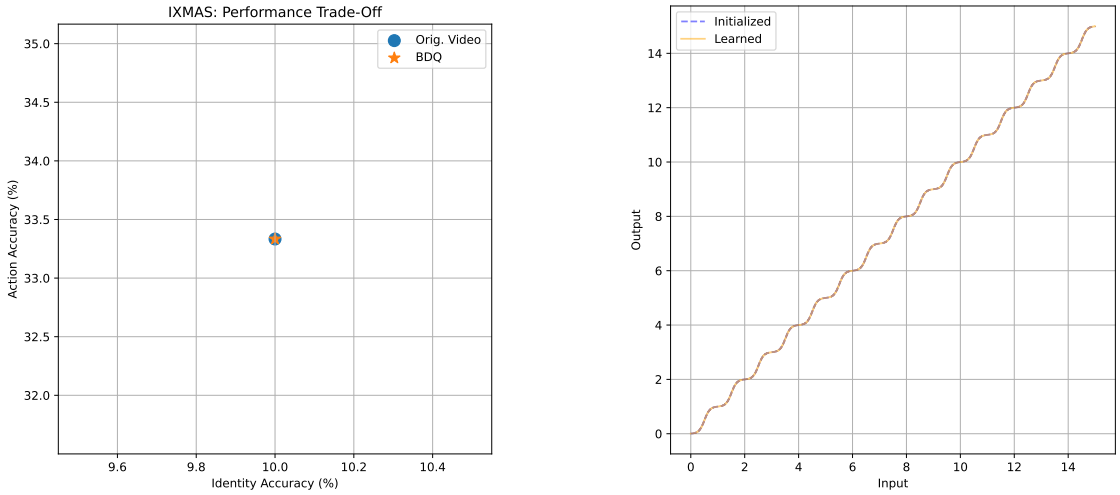
4 Results

todo

4.1 Results on KTH

4.2 Results on IXAMS

Update figures with results from latest runs.



(a) ?. (b) ?.

Figure 2: ?.

5 Ablation Study: Evaluation on the Privacy Attribute Prediction Module

"Ablation study with respect to BDQ encoder -> does the privacy attribute prediction module actually work?"

5.1 Experimental Setup

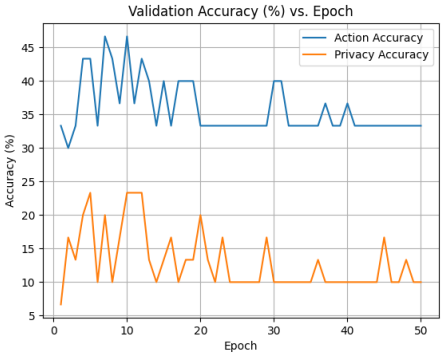
5.2 Results

Update figures with results from latest runs.

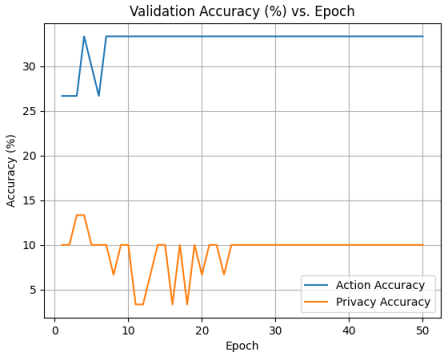
6 Discussion

todo

- Reproduction fidelity:



(a) ?.



(b) ?.

Figure 3: ?.

  - How closely do the results match the original BDQ?
  - Any discrepancies and insights

- Effectiveness on new data (IXAMS)
- Limitations and challenges (dataset differences, hardware constraints, etc.)

7 Contributions

The following is an overview of the division of work across team members.

Table 1: Individual Contributions.

Name	Main Contributions
Ciprian Bîcă	• Blur module • Privacy attribute prediction model • Training execution • Blog writing
Tadas Žaliauskas	• Difference module • Action recognition model • Adversarial training pipeline • Training execution
Yizhen Zang	• Quantization module • Dataset preprocessing pipeline • Additional dataset • Training execution • Blog writing
Žygis Liutkus	• Loss functions • Adversarial training pipeline • Validation pipeline • Training execution • Blog writing

References

[1] Z. W. Wang, V. Vineet, F. Pittaluga, S. Sinha, O. Cossairt, and S. B. Kang, "Privacy-Preserving Action Recognition using Coded Aperture Videos," Apr. 2019.

[2] T. Orekondy, B. Schiele, and M. Fritz, "Towards a Visual Privacy Advisor: Understanding and Predicting Privacy Risks in Images," Aug. 2017.

[3] S. Kumawat and H. Nagahara, "Privacy-Preserving Action Recognition via Motion Difference Quantization," Aug. 2022.

[4] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, (Cambridge, UK), pp. 32–36 Vol.3, IEEE, 2004.

[5] D. Weinland, R. Ronfard, and E. Boyer, "Free viewpoint action recognition using motion history volumes," *Computer Vision and Image Understanding*, vol. 104, pp. 249–257, Nov. 2006.