Documentation

Type	Technical Spec
Created	@October 22, 2024 6:06 PM
Last Edited Time	@October 22, 2024 8:21 PM
Last Edited By	Aman Agrawal @doubleCradle

`datagen.py`

Overview

`datagen.py` is a Python script designed to generate synthetic data by applying macroblocking artifacts to video frames. The script processes extracted frames from videos, applies random modifications to create artifacts, and saves the modified frames along with their parameters in a CSV file.

Dependencies

- `cv2` (OpenCV)
- `numpy`
- `os`
- `csv`
- `random`
- `tqdm` (for progress bars)

Functions

1. `apply_shift(block, shift_x, shift_y)`

Shifts a given block of pixels by specified amounts in the x and y directions.

Parameters:

- `block`: A numpy array representing the block of pixels to be shifted.
- `shift_x`: The amount to shift the block in the x direction.

- `shift_y`: The amount to shift the block in the y direction.

Returns:

- A numpy array representing the shifted block.

2. `create_artifacts_with_directional_copying(frame, block_size, max_shift_value, artifact_percentage, horizontal=True, start_from_top_or_left=True)`

Creates artifacts in a frame by copying content from unmodified areas and applying shifts.

Parameters:

- `frame`: A numpy array representing the original frame.
- `block_size`: The size of the blocks to be manipulated.
- `max_shift_value`: The maximum value for shifting the blocks.
- `artifact_percentage`: The percentage of blocks to be modified.
- `horizontal`: A boolean indicating whether to apply shifts horizontally (default: True).
- `start_from_top_or_left`: A boolean indicating whether to start from the top/left (default: True).

Returns:

- A numpy array representing the modified frame with artifacts.

3. `randomize_parameters(base_parameters)`

Randomizes parameters for block size, shift values, and artifact percentage based on provided base parameters.

Parameters:

- `base_parameters`: A dictionary containing the minimum and maximum values for block size, shift values, and artifact percentage.

Returns:

- A dictionary containing randomized parameters.

4. `process_extracted_frames(frames_folder, output_folder, base_parameters, num_variations)`

Processes extracted frames from a specified folder, applies random modifications, and saves the modified frames along with their parameters in a CSV file.

Parameters:

- `frames_folder`: The directory containing the extracted frames.
- `output_folder`: The directory where modified frames will be saved.
- `base_parameters`: A dictionary containing base parameters for randomization.
- `num_variations`: The number of variations to apply to each frame.

Returns:

- None (saves modified frames and parameters to a CSV file).

Usage

To use this script, ensure that the required dependencies are installed and that the configuration file (`configurations.py`) is set up correctly. The script is typically called from a main script (e.g., `main.py`) that handles video frame extraction and initiates the processing of frames.

Example

```
from datagen import process_extracted_frames
process_extracted_frames('path/to/extracted_frames', 'path/
to/output_frames', base_parameters, num_variations)
```

Output

- Modified frames are saved in the specified output folder.
- A CSV file (`dataset.csv`) is created in the `data` directory, containing the parameters used for each modified frame.

`configurations.py`

Overview

`configurations.py` is a configuration file that defines various parameters and settings used throughout the synthetic data generation process. It contains paths for input and output directories, as well as parameters for generating artifacts in video frames.

Configuration Dictionary

`DEMO_CONFIG`

A dictionary containing the following keys and their respective values:

- `data`:
    - Type: `str`
    - Description: The directory name where the dataset (including CSV file and modified frames) will be stored.
    - Example: `'dataset'`
- `input`:
    - Type: `str`
    - Description: The directory containing the input video files to be processed.
    - Example: `'videos'`
- `frames`:
    - Type: `str`
    - Description: The directory where extracted frames from the videos will be stored.
    - Example: `'dataset/extracted_frames'`
- `mod_frames`:
    - Type: `str`
    - Description: The directory where modified frames (with artifacts) will be saved.
    - Example: `'dataset/modified_frames'`
- `params`:

    - Type: `dict`
    - Description: A dictionary containing parameters for randomization during artifact generation.
    - Keys:
        - `block_size_min`: Minimum block size for artifacts (integer).
        - `block_size_max`: Maximum block size for artifacts (integer).
        - `max_shift_value_min`: Minimum shift value for blocks (integer).
        - `max_shift_value_max`: Maximum shift value for blocks (integer).
        - `artifact_percentage_min`: Minimum percentage of blocks to be modified (float).
        - `artifact_percentage_max`: Maximum percentage of blocks to be modified (float).
    - Example:

```
pythonCopy Code
'params': {
    'block_size_min': 10,
    'block_size_max': 50,
    'max_shift_value_min': 5,
    'max_shift_value_max': 20,
    'artifact_percentage_min': 0.1,
    'artifact_percentage_max': 0.7,
}
```

- `variations`:
    - Type: `int`
    - Description: The number of variations to apply to each extracted frame.
    - Example: `5`
- `useSeed`:
    - Type: `bool`

    - Description: A flag indicating whether to use a fixed seed for randomization (for reproducibility).
    - Example: `False`
- `seed`:
    - Type: `int`
    - Description: The seed value used for random number generation (only applicable if `useSeed` is `True`).
    - Example: `43`
- `fps`:
    - Type: `int`
    - Description: The frames per second (FPS) at which to extract frames from the input videos.
    - Example: `1`

Usage

This configuration file is imported and used by other scripts (e.g., `main.py` and `datagen.py`) to access the defined parameters and paths. It allows for easy adjustments to the settings without modifying the core logic of the code.

Example

To access a specific configuration value, you can import the `DEMO_CONFIG` dictionary as follows:

```
from configurations import DEMO_CONFIG

input_directory = DEMO_CONFIG['input']
```

`main.py`

Overview

`main.py` is the primary script responsible for defining logic for extracting frames from video files and processing those frames to generate synthetic data with macroblocking artifacts. It utilizes the configuration settings defined in `configurations.py` and the frame processing functions from `datagen.py`.

Dependencies

- `os`
- `glob`
- `tqdm` (for progress bars)
- `process_extracted_frames` from `datagen.py`
- `DEMO_CONFIG` from `configurations.py`

Script Workflow

1. **Extract Frames from Videos**:
    - The script searches for video files in the specified input directory.
    - It uses `ffmpeg` to extract frames from each video at a defined frames per second (FPS) rate.
    - Extracted frames are saved in the specified output directory.
2. **Process Extracted Frames**:
    - After extracting frames, the script calls the `process_extracted_frames` function to apply random modifications to the frames and save the results.

Configuration Variables

The script uses the following configuration variables from `DEMO_CONFIG`:

- **`input`**: Directory containing the input video files.
- **`frames`**: Directory where extracted frames will be saved.
- **`fps`**: Frames per second for frame extraction.
- **`mod_frames`**: Directory where modified frames will be saved.
- **`params`**: Base parameters for randomization.
- **`variations`**: Number of variations to apply to each frame.

Functions

Main Execution Flow

The script does not define any functions but executes the following steps in sequence:

1. **Set Up Directories**:
   - Creates the output directory for extracted frames if it does not exist.
2. **Find Video Files**:
   - Uses `glob` to find all video files in the specified input directory.
3. **Extract Frames**:
   - Iterates over each video file and constructs an `ffmpeg` command to extract frames.
   - Executes the command using `os.system()`.
4. **Process Extracted Frames**:
   - Calls `process_extracted_frames` to apply random modifications to the extracted frames.

Example

To run the script, simply execute it in a Python environment where the required dependencies are installed:

```
python main.py
```

Output

- Extracted frames are saved in the specified `frames` directory.
- Modified frames with artifacts are saved in the specified `mod_frames` directory.
- A CSV file (`dataset.csv`) is created in the `data` directory, containing the parameters used for each modified frame.