Tools

Tools for helping create neural networks

- Noise
- Toolkit

Noise

Module: `deeprai.tools.noise`

This module provides a set of classes for introducing different types of noise into numpy arrays, typically used for image data augmentation or robustness testing.

---

1. GaussianNoise Class

Description:

The `GaussianNoise` class applies Gaussian noise to a list of numpy arrays (images).

Attributes:

- **mean** (`float`, default=0): Mean of the Gaussian distribution.
- **std** (`float`, default=1): Standard deviation of the Gaussian distribution.

Methods:

- **compute()**: Internal method to get a function that introduces Gaussian noise to an image.
- **noise(arrays)**: Applies Gaussian noise to a list of numpy arrays. Uses multi-threading for efficiency.

Usage:

```
from deeprai.tools.noise import GaussianNoise


gaussian_noise = GaussianNoise(mean=0, std=25)
noisy_images = gaussian_noise.noise(list_of_images)
```

2. SaltPepperNoise Class

Description:

The `SaltPepperNoise` class introduces salt and pepper noise to a list of numpy arrays.

Attributes:

- **s_vs_p** (`float`, default=0.5): Proportion of salt vs. pepper noise.
- **amount** (`float`, default=0.04): Overall amount of noise to introduce.

Methods:

- **compute()**: Internal method to get a function that introduces salt and pepper noise to an image.
- **noise(arrays)**: Applies salt and pepper noise to a list of numpy arrays. Uses multi-threading for efficiency.

Usage:

```
from deeprai.tools.noise import SaltPepperNoise


sp_noise = SaltPepperNoise(s_vs_p=0.5, amount=0.04)
noisy_images = sp_noise.noise(list_of_images)
```

3. SpeckleNoise Class

Description:

The `SpeckleNoise` class introduces speckle noise to a list of numpy arrays.

Methods:

- **compute()**: Internal method to get a function that introduces speckle noise to an image.
- **noise(arrays)**: Applies speckle noise to a list of numpy arrays. Uses multi-threading for efficiency.

Usage:

```
from deeprai.tools.noise import SpeckleNoise

speckle_noise = SpeckleNoise()
noisy_images = speckle_noise.noise(list_of_images)
```

General Note:

For all the above classes, the `noise` method is designed for efficient computation by applying noise to multiple images using multi-threading. Each image in the input list is processed in a separate thread.

The results are then compiled and returned as a list of numpy arrays.

Toolkit

Module: `deeprai.tools.toolkit`

This module provides a collection of utility functions designed for numpy arrays. These functions offer various operations like verification, rounding, normalization, reshaping, and others, enhancing usability and information retrieval from numpy arrays.

---

1. `verify_inputs(array)`

Description:

Verify if the given input is a numpy array.

Parameters:

- **array**: The input to be checked.

Returns:

- **bool**: True if the input is a numpy array, otherwise False.

Example:

```
from deeprai.tools.toolkit import verify_inputs

result = verify_inputs(np.array([1, 2, 3]))
print(result)  # True
```

---

2. `round_out(array, a=2)`

Description:

Round the elements of a numpy array and set specific print options.

Parameters:

- **array** (`np.ndarray`): The input numpy array.
- **a** (`int`, optional): Decimal places to round to. Defaults to 2.

Returns:

- **np.ndarray**: The rounded numpy array.

Example:

```
from deeprai.tools.toolkit import round_out


rounded_array = round_out(np.array([1.12345, 2.6789]))
print(rounded_array)  # [1.12, 2.68]
```

---

3. `normalize(array)`

Description:

Normalize the elements of the numpy array to the range [0, 1].

Parameters:

- **array** (`np.ndarray`): The input array.

Returns:

- **np.ndarray**: The normalized array.

Example:

```
from deeprai.tools.toolkit import normalize


norm_array = normalize(np.array([10, 20, 30, 40]))
print(norm_array)
```

---

4. `reshape_to_2d(array)`

Description:

Reshape the numpy array to a 2D format if it's not already in that shape.

Parameters:

- **array** (`np.ndarray`): The input array.

Returns:

- **np.ndarray**: The reshaped 2D array.

Example:

```
from deeprai.tools.toolkit import reshape_to_2d

reshaped_array = reshape_to_2d(np.array([1, 2, 3, 4]))
print(reshaped_array)
```

---

5. `is_square_matrix(array)`

Description:

Check if the given numpy array is a square matrix.

Parameters:

- **array** (`np.ndarray`): The input array.

Returns:

- **bool**: True if the array is a square matrix, otherwise False.

Example:

```
from deeprai.tools.toolkit import is_square_matrix


result = is_square_matrix(np.array([[1, 2], [3, 4]]))
print(result)  # True
```

---

6. `sum_along_axis(array, axis=0)`

Description:

Compute the sum of elements of the numpy array along a specified axis.

Parameters:

- **array** (`np.ndarray`): The input array.
- **axis** (`int`, optional): Axis along which the sum is computed. Defaults to 0.

Returns:

- **np.ndarray**: The sum along the specified axis.

Example:

```
from deeprai.tools.toolkit import sum_along_axis


summed_array = sum_along_axis(np.array([[1, 2], [3, 4]]))
print(summed_array)  # [4, 6]
```

---

7. `array_info(array)`

Description:

Retrieve essential information about the numpy array.

Parameters:

- **array** (`np.ndarray`): The input array.

Returns:

- **dict**: A dictionary containing shape, data type, minimum and maximum values.

Example:

```
from deeprai.tools.toolkit import array_info


info = array_info(np.array([[1, 2], [3, 4]]))
print(info)
```