conv_layer_utils.py

This file contains all the code from conv_layer_utils.py

Code

```
from nndl.conv_layers import spatial_batchnorm_backward, spatial_batchnorm_forward
from nndl.layers import *
from utils.fast_layers import *


def conv_relu_forward(x, w, b, conv_param):
  """
  A convenience layer that performs a convolution followed by a ReLU.

  Inputs:
  - x: Input to the convolutional layer
  - w, b, conv_param: Weights and parameters for the convolutional layer

  Returns a tuple of:
  - out: Output from the ReLU
  - cache: Object to give to the backward pass
  """
  a, conv_cache = conv_forward_fast(x, w, b, conv_param)
  out, relu_cache = relu_forward(a)
  cache = (conv_cache, relu_cache)
  return out, cache


def conv_relu_backward(dout, cache):
  """
  Backward pass for the conv-relu convenience layer.
  """
  conv_cache, relu_cache = cache
  da = relu_backward(dout, relu_cache)
  dx, dw, db = conv_backward_fast(da, conv_cache)
  return dx, dw, db


def conv_relu_pool_forward(x, w, b, conv_param, pool_param):
  """
  Convenience layer that performs a convolution, a ReLU, and a pool.

  Inputs:
  - x: Input to the convolutional layer
```

```
  - w, b, conv_param: Weights and parameters for the convolutional layer
  - pool_param: Parameters for the pooling layer

  Returns a tuple of:
  - out: Output from the pooling layer
  - cache: Object to give to the backward pass
  """
  a, conv_cache = conv_forward_fast(x, w, b, conv_param)
  s, relu_cache = relu_forward(a)
  out, pool_cache = max_pool_forward_fast(s, pool_param)
  cache = (conv_cache, relu_cache, pool_cache)
  return out, cache


def conv_relu_pool_backward(dout, cache):
  """
  Backward pass for the conv-relu-pool convenience layer
  """
  conv_cache, relu_cache, pool_cache = cache
  ds = max_pool_backward_fast(dout, pool_cache)
  da = relu_backward(ds, relu_cache)
  dx, dw, db = conv_backward_fast(da, conv_cache)
  return dx, dw, db


def conv_relu_pool_batchnorm_forward(x, w, b, conv_param, pool_param, gamma, beta, bn_params
  """
  Convenience layer that performs a convolution, a ReLU, and a pool.

  Inputs:
  - x: Input to the convolutional layer
  - w, b, conv_param: Weights and parameters for the convolutional layer
  - pool_param: Parameters for the pooling layer

  Returns a tuple of:
  - out: Output from the pooling layer
  - cache: Object to give to the backward pass
  """
  a, conv_cache = conv_forward_fast(x, w, b, conv_param)
  b, bn_cache = spatial_batchnorm_forward(a,gamma,beta,bn_params)
  s, relu_cache = relu_forward(a)
  out, pool_cache = max_pool_forward_fast(s, pool_param)
  cache = (conv_cache,bn_cache, relu_cache, pool_cache)
  return out, cache
```

```
def conv_relu_pool_batchnorm_backward(dout, cache):
  """
  Backward pass for the conv-relu-pool convenience layer
  """
  conv_cache,bn_cache, relu_cache, pool_cache = cache
  ds = max_pool_backward_fast(dout, pool_cache)
  da = relu_backward(ds, relu_cache)
  dc,dgamma,dbeta = spatial_batchnorm_backward(da,bn_cache)
  dx, dw, db = conv_backward_fast(dc, conv_cache)
  return dx, dw, db,dgamma,dbeta
```