cktran_09859713

October 6, 2021

1 EECS 442 PS4: Backpropagation

Calvin Tran, cktran

2 Starting

Run the following code to import the modules you'll need. After your finish the assignment, remember to run all cells and save the note book to your local machine as a .ipynb file for Canvas submission.

[1]:
```
import pickle
import numpy as np
import matplotlib.pyplot as plt
import os
import math
from torchvision.datasets import CIFAR10
download = not os.path.isdir('cifar-10-batches-py')
dset_train = CIFAR10(root='.', download=download)
```

3 Problem 4.1 Understanding Backpropagation

4 4.1 (b)

Implement the code for forward and backward pass of computation graph in (a)

[2]:
```
def f_1(x0, x1, w0, w1, w2):
    """
    Computes the forward and backward pass through the computational graph
    of (a)

    Inputs:
    - x0, x1, w0, w1, w2: Python floats

    Returns a tuple of:
    - L: The output of the graph
    - grads: A tuple (grad_x0, grad_x1, grad_w0, grad_w1, grad_w2)
    giving the derivative of the output L with respect to each input.
```

```
    """
    ##############################################################################
    # TODO: Implement the forward pass for the computational graph for (a) and#
    # store the output of this graph as L                                        #
    ##############################################################################
    a = w0 * x0
    b = w1 * x1
    p = a + b
    c = p + w2
    n = c * (-1)
    e = exp(n)
    d = e + 1
    L = 1/d
    ##############################################################################
    #                                END OF YOUR CODE                            #
    ##############################################################################



    ##############################################################################
    # TODO: Implement the backward pass for the computational graph for (a)   #
    # Store the gradients for each input                                         #
    ##############################################################################
    grad_L = 1
    grad_d = -1/d**2
    grad_e = grad_L
    grad_n = e
    grad_c = -1
    grad_p = grad_L
    grad_a = grad_L
    grad_b = grad_L

    grad_x0 = grad_d * grad_e * grad_n * grad_c * grad_p * grad_a * w_0
    grad_x1 = grad_d * grad_e * grad_n * grad_c * grad_p * grad_b * w_1
    grad_w0 = grad_d * grad_e * grad_n * grad_c * grad_p * grad_a * x_0
    grad_w1 = grad_d * grad_e * grad_n * grad_c * grad_p * grad_b * x_1
    grad_w2 = grad_d * grad_e * grad_n * grad_c * grad_L
    ##############################################################################
    #                                END OF YOUR CODE                            #
    ##############################################################################

    grads = (grad_x0, grad_x1, grad_w0, grad_w1, grad_w2)
    return L, grads
```

5 4.1 (c)

Implement the code for forward and backward pass of computation graph in (c)

[3]:
```
def f_2(x, y, z):
    """
    Computes the forward and backward pass through the computational graph
    of (c)

    Inputs:
    - x, y, z: Python floats

    Returns a tuple of:
    - L: The output of the graph
    - grads: A tuple (grad_x, grad_y, grad_z)
    giving the derivative of the output L with respect to each input.
    """
    ############################################################################
    # TODO: Implement the forward pass for the computational graph for (c) and#
    # store the output of this graph as L                                      #
    ############################################################################
    a = x * (-1)
    b = exp(y)
    b1 = b
    b2 = b
    c = exp(z)
    c1 = c
    c2 = c
    p = b1 * c1
    p1 = p
    p2 = p
    d = b2 + p1
    e = c2 / p2
    m = a - d
    n = m / e
    L = n**2
    ############################################################################
    #                             END OF YOUR CODE                             #
    ############################################################################



    ############################################################################
    # TODO: Implement the backward pass for the computational graph for (c)    #
    # Store the gradients for each input                                       #
    ############################################################################
    grad_L = 1
    grad_n = 2 * n
```

```
    grad_m = grad_L / e
    grad_e = -n / e
    grad_d = -1

    grad_p1 = grad_L
    grad_p2 = -e / p2

    grad_a = grad_L

    grad_b1 = grad_L * c1
    grad_b2 = grad_L

    grad_c1 = grad_L * b1

    grad_x = grad_n * grad_m * grad_a * -1
    grad_y = grad_n * ((grad_m * grad_d)(grad_p1 * grad_b1 * y + grad_b2 * y) +␣
↪(grad_e * grad_p2 * grad_b1 * y))
    grad_z = grad_n * grad_m * grad_d * grad_p1 * grad_c1 * z
    ##############################################################################
    #                                END OF YOUR CODE                            #
    ##############################################################################

    grads = (grad_x, grad_y, grad_z)
    return L, grads
```

6 Problem 4.2 Softmax Classifier with Two Layer Neural Network

In this problem you will develop a two Layer neural network with fully-connected layers to perform classification, and test it out on the CIFAR-10 dataset.

We train the network with a softmax loss function on the weight matrices. The network uses a ReLU nonlinearity after the first fully connected layer. In other words, the network has the following architecture:

input - fully connected layer - ReLU - fully connected layer - softmax

The outputs of the second fully-connected layer are the scores for each class.

You cannot use any deep learning libraries such as PyTorch in this part.

7 4.2 (a) Layers

In this problem, implement fully connected layer, relu. Softmax layer has already been implemented in the provided code. Filling in all TODOs in skeleton codes will be sufficient.

[4]:
```
def fc_forward(X, W, b):
    """
    Computes the forward pass for a fully-connected layer.

    The input X has shape (N, Din) and contains a minibatch of N
```

```
    examples, where each example x[i] has shape (Din,).

    Inputs:
    - X: A numpy array containing input data, of shape (N, Din)
    - W: A numpy array of weights, of shape (Din, Dout)
    - b: A numpy array of biases, of shape (Dout,)

    Returns a tuple of:
    - out: output, of shape (N, Dout)
    - cache: (X, W, b)
    """
    ###########################################################################
    # TODO: Implement the forward pass. Store the result in out.              #
    ###########################################################################

    out = X.dot(W) + b

    ###########################################################################
    #                             END OF YOUR CODE                            #
    ###########################################################################
    cache = (X, W, b)
    return out, cache


def fc_backward(dout, cache):
    """
    Computes the backward pass for a fully_connected layer.

    Inputs:
    - dout: Upstream derivative, of shape (N, Dout)
    - cache: returned by your forward function. Tuple of:
      - X: Input data, of shape (N, Din)
      - W: Weights, of shape (Din, Dout)
      - b: Biases, of shape (Dout,)

    Returns a tuple of:
    - dX: Gradient with respect to X, of shape (N, Din)
    - dW: Gradient with respect to W, of shape (Din, Dout)
    - db: Gradient with respect to b, of shape (Dout,)
    """
    X, W, b = cache
    dX, dW, db = None, None, None
    ###########################################################################
    # TODO: Implement the affine backward pass.                               #
    ###########################################################################

    dX = dout.dot(W.T)
```

```
    dW = X.T.dot(dout)
    db = np.sum(dout, axis=0)

    ###########################################################################
    #                             END OF YOUR CODE                            #
    ###########################################################################
    return dX, dW, db

def relu_forward(x):
    """
    Computes the forward pass for a layer of rectified linear units (ReLUs).

    Input:
    - x: Inputs, of any shape

    Returns a tuple of:
    - out: Output, of the same shape as x
    - cache: x
    """
    out = x.copy()
    ###########################################################################
    # TODO: Implement the ReLU forward pass.                                  #
    ###########################################################################

    out = np.maximum(0, out)

    ###########################################################################
    #                             END OF YOUR CODE                            #
    ###########################################################################
    cache = x
    return out, cache


def relu_backward(dout, cache):
    """
    Computes the backward pass for a layer of rectified linear units (ReLUs).

    Input:
    - dout: Upstream derivatives, of any shape
    - cache: returned by your forward function. Input x, of same shape as dout

    Returns:
    - dx: Gradient with respect to x
    """
    dx, x = dout.copy(), cache
    ###########################################################################
    # TODO: Implement the ReLU backward pass.                                 #
```

```
    ###########################################################################

    dx[x <= 0] = 0

    ###########################################################################
    #                             END OF YOUR CODE                            #
    ###########################################################################
    return dx


def softmax_loss(X, y):
    """
    Computes the loss and gradient for softmax classification.

    Inputs:
    - X: Input data, of shape (N, C) where x[i, j] is the score for the jth
      class for the ith input.
    - y: Vector of labels, of shape (N,) where y[i] is the label for X[i] and
      0 <= y[i] < C

    Returns a tuple of:
    - loss: Scalar giving the loss
    - dX: Gradient of the loss with respect to x
    """
    loss, dX = None, None                                             #
    dX = np.exp(X - np.max(X, axis=1, keepdims=True))
    dX /= np.sum(dX, axis=1, keepdims=True)
    loss = -np.sum(np.log(dX[np.arange(X.shape[0]), y])) / X.shape[0]
    dX[np.arange(X.shape[0]), y] -= 1
    dX /= X.shape[0]

    return loss, dX
```

8 4.2 (b) Softmax Classifier

In this problem, implement softmax classifier.

[5]:
```
class SoftmaxClassifier(object):
    """
    A fully-connected neural network with
    softmax loss that uses a modular layer design. We assume an input dimension
    of D, a hidden dimension of H, and perform classification over C classes.

    The architecture should be fc - relu - fc - softmax with one hidden layer

    The learnable parameters of the model are stored in the dictionary
    self.params that maps parameter names to numpy arrays.
```

```
    """

    def __init__(self, input_dim=3072, hidden_dim=300, num_classes=10,
                 weight_scale=1e-3):
        """
        Initialize a new network.

        Inputs:
        - input_dim: An integer giving the size of the input
        - hidden_dim: An integer giving the size of the hidden layer, None
          if there's no hidden layer.
        - num_classes: An integer giving the number of classes to classify
        - weight_scale: Scalar giving the standard deviation for random
          initialization of the weights.
        """
        self.params = {}
        
############################################################################
        # TODO: Initialize the weights and biases of the two-layer net. Weights    #
        # should be initialized from a Gaussian centered at 0.0 with               #
        # standard deviation equal to weight_scale, and biases should be           #
        # initialized to zero. All weights and biases should be stored in the      #
        # dictionary self.params, with fc weights and biases using the keys        #
        # 'W' and 'b', i.e., W1, b1 for the weights and bias in the first linear   #
        # layer, W2, b2 for the weights and bias in the second linear layer.       #
        
############################################################################

        self.params['W1'] = np.random.normal(scale=weight_scale, size=(input_dim, hidden_dim))
        self.params['b1'] = np.zeros(hidden_dim)
        self.params['W2'] = np.random.normal(scale=weight_scale, size=(hidden_dim, num_classes))
        self.params['b2'] = np.zeros(num_classes)

        
############################################################################
        #                             END OF YOUR CODE                             #
        
```

```
    ␣
↪#############################################################################


  def forwards_backwards(self, X, y=None):
      """
      Compute loss and gradient for a minibatch of data.

      Inputs:
      - X: Array of input data of shape (N, Din)
      - y: Array of labels, of shape (N,). y[i] gives the label for X[i].

      Returns:
      If y is None, then run a test-time forward pass of the model and return:
      - scores: Array of shape (N, C) giving classification scores, where
        scores[i, c] is the classification score for X[i] and class c.

      If y is not None, then run a training-time forward and backward pass.␣
↪And
      return a tuple of:
      - loss: Scalar value giving the loss
      - grads: Dictionary with the same keys as self.params, mapping␣
↪parameter
        names to gradients of the loss with respect to those parameters.
      """
      scores = None
    ␣
↪#############################################################################
      # TODO: Implement the forward pass for the two-layer net, computing the␣
↪   #
      # class scores for X and storing them in the scores variable.          ␣
↪   #
    ␣
↪#############################################################################

      out_1, cache_1 = fc_forward(X, self.params['W1'], self.params['b1'])
      relu_out, relu_cache = relu_forward(out_1)
      out_2, cache_2 = fc_forward(relu_out, self.params['W2'], self.
↪params['b2'])
      scores = out_2


    ␣
↪#############################################################################
      #                             END OF YOUR CODE                         ␣
↪   #
```

```
        ␣
↪#############################################################################

        # If y is None then we are in test mode so just return scores
        if y is None:
            return scores

        loss, grads = 0, {}
        ␣
↪#############################################################################
        # TODO: Implement the backward pass for the two-layer net. Store the␣
↪loss  #
        # in the loss variable and gradients in the grads dictionary. Compute␣
↪data #
        # loss using softmax, and make sure that grads[k] holds the gradients␣
↪for  #
        # self.params[k].                                                      ␣
↪    #
        ␣
↪#############################################################################

        loss, dscores = softmax_loss(scores, y)

        dx_2, grads['W2'], grads['b2'] = fc_backward(dscores, cache_2)
        dhidden = relu_backward(dx_2, relu_cache)
        dx_1, grads['W1'], grads['b1'] = fc_backward(dhidden, cache_1)

        ␣
↪#############################################################################
        #                             END OF YOUR CODE                         ␣
↪    #
        ␣
↪#############################################################################
        return loss, grads
```

9 4.2(c) Training

In this problem, you need to preprocess the images and set up model hyperparameters. Notice that adjust the training and val split is optional.

[8]:
```
def unpickle(file):
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding="latin1")
    return dict
```

```
def load_cifar10():
    data = {}
    meta = unpickle("cifar-10-batches-py/batches.meta")
    batch1 = unpickle("cifar-10-batches-py/data_batch_1")
    batch2 = unpickle("cifar-10-batches-py/data_batch_2")
    batch3 = unpickle("cifar-10-batches-py/data_batch_3")
    batch4 = unpickle("cifar-10-batches-py/data_batch_4")
    batch5 = unpickle("cifar-10-batches-py/data_batch_5")
    test_batch = unpickle("cifar-10-batches-py/test_batch")
    X_train = np.vstack((batch1['data'], batch2['data'], batch3['data'],\
                         batch4['data'], batch5['data']))
    Y_train = np.array(batch1['labels'] + batch2['labels'] + batch3['labels'] +
                       batch4['labels'] + batch5['labels'])
    X_test = test_batch['data']
    Y_test = test_batch['labels']

    #Preprocess images here
    X_train = (X_train-np.mean(X_train,axis=1,keepdims=True))/np.
 ↪std(X_train,axis=1,keepdims=True)
    X_test = (X_test-np.mean(X_test,axis=1,keepdims=True))/np.
 ↪std(X_test,axis=1,keepdims=True)

    data['X_train'] = X_train[:40000]
    data['y_train'] = Y_train[:40000]
    data['X_val'] = X_train[40000:]
    data['y_val'] = Y_train[40000:]
    data['X_test'] = X_test
    data['y_test'] = Y_test
    return data

def testNetwork(model, X, y, num_samples=None, batch_size=100):
    """
    Check accuracy of the model on the provided data.

    Inputs:
    - model: Image classifier
    - X: Array of data, of shape (N, d_1, ..., d_k)
    - y: Array of labels, of shape (N,)
    - num_samples: If not None, subsample the data and only test the model
      on num_samples datapoints.
    - batch_size: Split X and y into batches of this size to avoid using
      too much memory.

    Returns:
    - acc: Scalar giving the fraction of instances that were correctly
      classified by the model.
```

```
    """

    # Subsample the data
    N = X.shape[0]
    if num_samples is not None and N > num_samples:
        mask = np.random.choice(N, num_samples)
        N = num_samples
        X = X[mask]
        y = y[mask]

    # Compute predictions in batches
    num_batches = N // batch_size
    if N % batch_size != 0:
        num_batches += 1
    y_pred = []
    for i in range(num_batches):
        start = i * batch_size
        end = (i + 1) * batch_size
        scores = model.forwards_backwards(X[start:end])
        y_pred.append(np.argmax(scores, axis=1))
    y_pred = np.hstack(y_pred)
    acc = np.mean(y_pred == y)

    return acc

def SGD(W,dW, learning_rate=1e-3):
    """ Apply a gradient descent step on weight W
    Inputs:
        W : Weight matrix
        dW : gradient of weight, same shape as W
        learning_rate : Learning rate. Defaults to 1e-3.
    Returns:
        new_W: Updated weight matrix
    """

    # Apply a gradient descent step on weight W using the gradient dW and the␣
 ↪specified learning rate.
    new_W = W - learning_rate * dW

    return new_W

def trainNetwork(model, data, **kwargs):
    """
     Required arguments:
    - model: Image classifier
    - data: A dictionary of training and validation data containing:
      'X_train': Array, shape (N_train, d_1, ..., d_k) of training images
```

```
    'X_val': Array, shape (N_val, d_1, ..., d_k) of validation images
    'y_train': Array, shape (N_train,) of labels for training images
    'y_val': Array, shape (N_val,) of labels for validation images

  Optional arguments:
  - learning_rate: A scalar for initial learning rate.
  - lr_decay: A scalar for learning rate decay; after each epoch the
    learning rate is multiplied by this value.
  - batch_size: Size of minibatches used to compute loss and gradient
    during training.
  - num_epochs: The number of epochs to run for during training.
  - print_every: Integer; training losses will be printed every
    print_every iterations.
  - verbose: Boolean; if set to false then no output will be printed
    during training.
  - num_train_samples: Number of training samples used to check training
    accuracy; default is 1000; set to None to use entire training set.
  - num_val_samples: Number of validation samples to use to check val
    accuracy; default is None, which uses the entire validation set.
  - optimizer: Choice of using either 'SGD' or 'SGD_Momentum' for updating␣
↪weights; default is SGD.
  """


  learning_rate =  kwargs.pop('learning_rate', 1e-3)
  lr_decay = kwargs.pop('lr_decay', 1.0)
  batch_size = kwargs.pop('batch_size', 100)
  num_epochs = kwargs.pop('num_epochs', 10)
  num_train_samples = kwargs.pop('num_train_samples', 1000)
  num_val_samples = kwargs.pop('num_val_samples', None)
  print_every = kwargs.pop('print_every', 10)
  verbose = kwargs.pop('verbose', True)
  optimizer = kwargs.pop('optimizer', 'SGD')

  epoch = 0
  best_val_acc = 0
  best_params = {}
  loss_history = []
  train_acc_history = []
  val_acc_history = []


  num_train = data['X_train'].shape[0]
  iterations_per_epoch = max(num_train // batch_size, 1)
  num_iterations = num_epochs * iterations_per_epoch

  #Initialize velocity dictionary if optimizer is SGD_Momentum
```

```
  if optimizer == 'SGD_Momentum':
    velocity_dict = {p:np.zeros(w.shape) for p,w in model.params.items()}

  for t in range(num_iterations):
      # Make a minibatch of training data
      batch_mask = np.random.choice(num_train, batch_size)
      X_batch = data['X_train'][batch_mask]
      y_batch = data['y_train'][batch_mask]

      # Compute loss and gradient
      loss, grads = model.forwards_backwards(X_batch, y_batch)
      loss_history.append(loss)

      # Perform a parameter update
      if optimizer == 'SGD':
        for p, w in model.params.items():
            model.params[p] = SGD(w,grads[p], learning_rate=learning_rate)

      elif optimizer == 'SGD_Momentum':
        for p, w in model.params.items():
            model.params[p], velocity_dict[p] = SGD_Momentum(w, grads[p],␣
↪velocity_dict[p], beta=0.5, learning_rate=learning_rate)
      else:
        raise NotImplementedError
      # Print training loss
      if verbose and t % print_every == 0:
          print('(Iteration %d / %d) loss: %f' % (
                 t + 1, num_iterations, loss_history[-1]))

      # At the end of every epoch, increment the epoch counter and decay
      # the learning rate.
      epoch_end = (t + 1) % iterations_per_epoch == 0
      if epoch_end:
          epoch += 1
          learning_rate *= lr_decay

      # Check train and val accuracy on the first iteration, the last
      # iteration, and at the end of each epoch.
      first_it = (t == 0)
      last_it = (t == num_iterations - 1)
      if first_it or last_it or epoch_end:
          train_acc = testNetwork(model, data['X_train'], data['y_train'],
              num_samples= num_train_samples)
          val_acc = testNetwork(model, data['X_val'], data['y_val'],
              num_samples=num_val_samples)
          train_acc_history.append(train_acc)
          val_acc_history.append(val_acc)
```

```
            if verbose:
                print('(Epoch %d / %d) train acc: %f; val_acc: %f' % (
                       epoch, num_epochs, train_acc, val_acc))

            # Keep track of the best model
            if val_acc > best_val_acc:
                best_val_acc = val_acc
                best_params = {}
                for k, v in model.params.items():
                    best_params[k] = v.copy()

    model.params = best_params

    return model, train_acc_history, val_acc_history


# load data
data = load_cifar10()
train_data = { k: data[k] for k in ['X_train', 'y_train',
                                    'X_val', 'y_val']}
################################################################################
# TODO: Set up model hyperparameters for SGD                                   #
################################################################################

# initialize model
model_SGD = SoftmaxClassifier(hidden_dim = 300, weight_scale=1e-2)

# start training using SGD
model_SGD, train_acc_history_SGD, val_acc_history_SGD = trainNetwork(
    model_SGD, train_data, learning_rate = 1e-2,
    lr_decay=1, num_epochs=10,
    batch_size=100, print_every=1000, optimizer = 'SGD')
################################################################################
#                           END OF YOUR CODE                                   #
################################################################################
```

```
(Iteration 1 / 4000) loss: 2.323817
(Epoch 0 / 10) train acc: 0.098000; val_acc: 0.096400
(Epoch 1 / 10) train acc: 0.409000; val_acc: 0.363900
(Epoch 2 / 10) train acc: 0.440000; val_acc: 0.408800
(Iteration 1001 / 4000) loss: 1.509000
(Epoch 3 / 10) train acc: 0.459000; val_acc: 0.433100
(Epoch 4 / 10) train acc: 0.451000; val_acc: 0.449000
(Epoch 5 / 10) train acc: 0.494000; val_acc: 0.461500
(Iteration 2001 / 4000) loss: 1.511527
(Epoch 6 / 10) train acc: 0.538000; val_acc: 0.465700
```

```
(Epoch 7 / 10) train acc: 0.516000; val_acc: 0.476500
(Iteration 3001 / 4000) loss: 1.081032
(Epoch 8 / 10) train acc: 0.528000; val_acc: 0.489900
(Epoch 9 / 10) train acc: 0.546000; val_acc: 0.499300
(Epoch 10 / 10) train acc: 0.604000; val_acc: 0.499200
```

10 4.2(d) Training with SGD_Momentum

The model above was trained using SGD. Now implement the SGD_Momentum function to train the model using SGD with momentum.

[11]:
```
def SGD_Momentum(W, dW, velocity, beta=0.5, learning_rate=1e-3):
    """ Apply a gradient descent with momentum update on weight W
    Inputs:
        W : Weight matrix
        dW : gradient of weight, same shape as W
        velocity : velocity matrix, same shape as W
        beta : scalar value in range [0,1] weighting the velocity matrix.␣
 ↪Setting it to 0 should make SGD_Momentum same as SGD.
              Defaults to 0.5.
        learning_rate : Learning rate. Defaults to 1e-3.
    Returns:
        new_W: Updated weight matrix
        new_velocity: Updated velocity matrix
    """
    # ===== your code here! =====
    # TODO:
    # Apply a gradient descent step on weight W using the gradient dW and the␣
 ↪specified learning rate.
    # 1. Calculate the new velocity by using the velocity of last iteration␣
 ↪(input velocity) and gradient
    # 2. Update the weights using the new_velocity

    new_velocity = dW + beta * velocity
    new_W = W - learning_rate * velocity

    # ==== end of code ====
    return new_W, new_velocity

################################################################################
# TODO: Set up model hyperparameters for SGD_Momentum
# Your hyperparameters should be identical to what you used for SGD (without␣
 ↪momentum)#
################################################################################

# initialize model
model_SGD_Momentum = SoftmaxClassifier(hidden_dim = 300, weight_scale=1e-2)
```

```
# start training
#Using SGD_Momentum as optimizer for trainning for training
model_SGD_Momentum, train_acc_history_SGD_Momentum,␣
 ↪val_acc_history_SGD_Momentum = trainNetwork(
    model_SGD_Momentum, train_data, learning_rate = 1e-2,
    lr_decay=1, num_epochs=10,
    batch_size=100, print_every=1000, optimizer = 'SGD_Momentum')
################################################################################
#                             END OF YOUR CODE                                 #
################################################################################
```

```
(Iteration 1 / 4000) loss: 2.292672
(Epoch 0 / 10) train acc: 0.130000; val_acc: 0.120300
(Epoch 1 / 10) train acc: 0.407000; val_acc: 0.398900
(Epoch 2 / 10) train acc: 0.498000; val_acc: 0.445700
(Iteration 1001 / 4000) loss: 1.598398
(Epoch 3 / 10) train acc: 0.521000; val_acc: 0.462700
(Epoch 4 / 10) train acc: 0.521000; val_acc: 0.477500
(Epoch 5 / 10) train acc: 0.573000; val_acc: 0.494800
(Iteration 2001 / 4000) loss: 1.180691
(Epoch 6 / 10) train acc: 0.558000; val_acc: 0.499100
(Epoch 7 / 10) train acc: 0.614000; val_acc: 0.502200
(Iteration 3001 / 4000) loss: 1.298302
(Epoch 8 / 10) train acc: 0.610000; val_acc: 0.507700
(Epoch 9 / 10) train acc: 0.649000; val_acc: 0.512300
(Epoch 10 / 10) train acc: 0.634000; val_acc: 0.507000
```

11 4.2(e) Report Accuracy

Run the given code and report the accuracy of model_SGD and model_SGD_Momentum on test set. Which model trains more quickly? Is the ultimate validation accuracy different? Report your observation in the text block below.

[12]:
```
# report test accuracy
acc = testNetwork(model_SGD, data['X_test'], data['y_test'])
print("Test accuracy of model_SGD: {}".format(acc))
# report test accuracy
acc = testNetwork(model_SGD_Momentum, data['X_test'], data['y_test'])
print("Test accuracy of model_SGD_Momentum: {}".format(acc))
```

```
Test accuracy of model_SGD: 0.4983
Test accuracy of model_SGD_Momentum: 0.506
```

My observation:

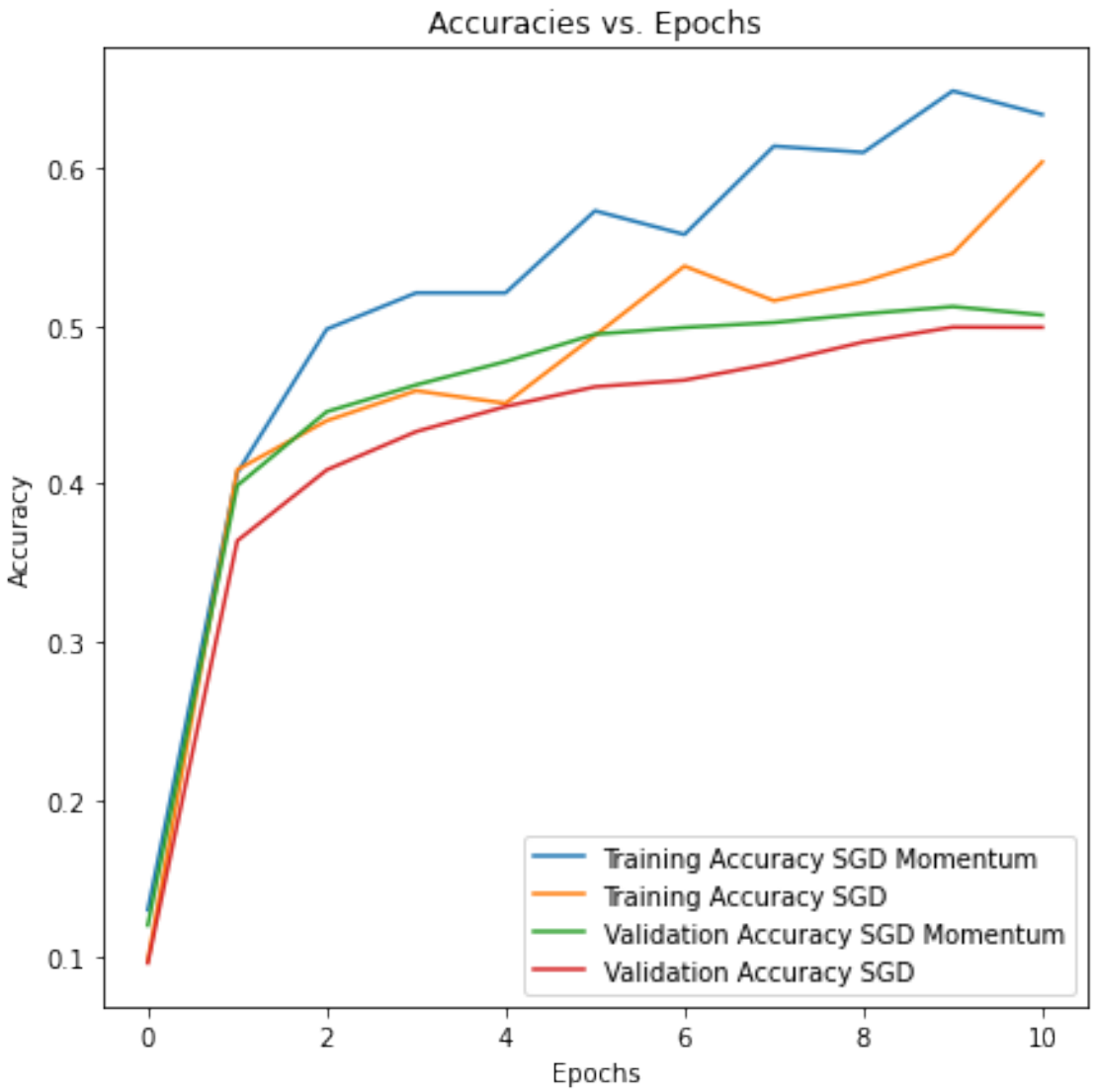
12 4.2(f) Plot

Using the train_acc_history and val_acc_history, plot the train & val accuracy versus epochs on one plot, using SGD and SGD_Momentum as optimizer.

[24]:
```
################################################################################
# Your Code here
################################################################################
f = plt.figure()
f.set_figwidth(7)
f.set_figheight(7)
plt.title("Accuracies vs. Epochs")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")

plt.plot(train_acc_history_SGD_Momentum, label='Training Accuracy SGD Momentum')
plt.legend()
plt.plot(train_acc_history_SGD, label='Training Accuracy SGD')
plt.legend()
plt.plot(val_acc_history_SGD_Momentum, label='Validation Accuracy SGD Momentum')
plt.legend()
plt.plot(val_acc_history_SGD, label='Validation Accuracy SGD')
plt.legend()

plt.show()
################################################################################
#                            END OF YOUR CODE                                  #
################################################################################
```



13 Convert Notebook to PDF

[ ]:
```
# generate pdf
# %%capture
!git clone https://gist.github.com/bc5f1add34fef7c7f9fb83d3783311e2.git
!cp bc5f1add34fef7c7f9fb83d3783311e2/colab_pdf.py colab_pdf.py
from colab_pdf import colab_pdf
# change the name to your ipynb file name shown on the top left of Colab window
# Important: make sure that your file name does not contain spaces!
colab_pdf('cktran_09859713.ipynb')
```

```
Cloning into 'bc5f1add34fef7c7f9fb83d3783311e2'...
```

```
remote: Enumerating objects: 10, done.
remote: Total 10 (delta 0), reused 0 (delta 0), pack-reused 10
Unpacking objects: 100% (10/10), done.
Mounted at /content/drive/
Get:1 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic InRelease
[15.9 kB]
Hit:2 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease
[3,626 B]
Hit:5 http://ppa.launchpad.net/cran/libgit2/ubuntu bionic InRelease
Get:6 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic InRelease [15.9 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:8 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease
Get:9 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Ign:10
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64
InRelease
Ign:11 https://developer.download.nvidia.com/compute/machine-
learning/repos/ubuntu1804/x86_64  InRelease
Hit:12
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64
Release
Hit:13 https://developer.download.nvidia.com/compute/machine-
learning/repos/ubuntu1804/x86_64  Release
Get:14 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main Sources
[1,802 kB]
Get:15 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main amd64
Packages [922 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages
[2,209 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages
[2,801 kB]
Get:18 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic/main amd64 Packages
[40.8 kB]
Get:20 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages
[1,430 kB]
Get:22 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages
[2,365 kB]
Fetched 11.9 MB in 4s (3,302 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
37 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

```
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-latex-base texlive-latex-extra
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
Suggested packages:
  fonts-noto apache2 | lighttpd | httpd poppler-utils ghostscript
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri
  ruby-dev bundler debhelper gv | postscript-viewer perl-tk xpdf-reader
  | pdf-viewer texlive-fonts-recommended-doc texlive-latex-base-doc
  python-pygments icc-profiles libfile-which-perl
  libspreadsheet-parseexcel-perl texlive-latex-extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex ruby-tcltk
  | libtcltk-ruby texlive-pictures-doc vprerex
The following NEW packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-generic-recommended
  texlive-latex-base texlive-latex-extra texlive-latex-recommended
  texlive-pictures texlive-plain-generic texlive-xetex tipa
0 upgraded, 47 newly installed, 0 to remove and 37 not upgraded.
Need to get 146 MB of archives.
After this operation, 460 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-droid-fallback
all 1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2
[2,698 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 poppler-data all
0.4.8-2 [1,479 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/main amd64 tex-common all 6.09
[33.0 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lmodern all
2.004.5-3 [4,551 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-noto-mono all
20171026-2 [75.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-texgyre all
20160520-1 [8,761 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/main amd64 javascript-common all
```

```
11 [6,066 B]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsfilters1
amd64 1.20.2-0ubuntu3.1 [108 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsimage2
amd64 2.2.7-1ubuntu2.8 [18.6 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/main amd64 libijs-0.35 amd64
0.35-13 [15.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjbig2dec0 amd64
0.13-6 [55.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9-common
all 9.26~dfsg+0-0ubuntu0.18.04.14 [5,092 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9 amd64
9.26~dfsg+0-0ubuntu0.18.04.14 [2,265 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all
3.2.1-1 [152 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libkpathsea6
amd64 2017.20170613.44572-8ubuntu0.1 [54.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic/main amd64 libpotrace0 amd64
1.14-2 [17.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libptexenc1
amd64 2017.20170613.44572-8ubuntu0.1 [34.5 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration
all 1.11 [4,994 B]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 ruby2.5 amd64
2.5.1-1ubuntu1.10 [48.6 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby amd64 1:2.5.1
[5,712 B]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 rake all
12.3.1-1ubuntu0.1 [44.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-did-you-mean all
1.2.0-2 [9,700 B]
Get:24 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-minitest all
5.10.3-1 [38.6 kB]
Get:25 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:26 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-power-assert all
0.3.0-1 [7,952 B]
Get:27 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-test-unit all
3.2.5-1 [61.1 kB]
Get:28 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libruby2.5
amd64 2.5.1-1ubuntu1.10 [3,071 kB]
Get:29 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libsynctex1
amd64 2017.20170613.44572-8ubuntu0.1 [41.4 kB]
Get:30 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexlua52
amd64 2017.20170613.44572-8ubuntu0.1 [91.2 kB]
Get:31 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexluajit2
amd64 2017.20170613.44572-8ubuntu0.1 [230 kB]
Get:32 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libzzip-0-13
```

```
amd64 0.13.62-3.1ubuntu0.18.04.1 [26.0 kB]
Get:33 http://archive.ubuntu.com/ubuntu bionic/main amd64 lmodern all 2.004.5-3
[9,631 kB]
Get:34 http://archive.ubuntu.com/ubuntu bionic/main amd64 preview-latex-style
all 11.91-1ubuntu1 [185 kB]
Get:35 http://archive.ubuntu.com/ubuntu bionic/main amd64 t1utils amd64 1.41-2
[56.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tex-gyre all
20160520-1 [4,998 kB]
Get:37 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 texlive-
binaries amd64 2017.20170613.44572-8ubuntu0.1 [8,179 kB]
Get:38 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-base all
2017.20180305-1 [18.7 MB]
Get:39 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-fonts-
recommended all 2017.20180305-1 [5,262 kB]
Get:40 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-plain-
generic all 2017.20180305-2 [23.6 MB]
Get:41 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-generic-
recommended all 2017.20180305-1 [15.9 kB]
Get:42 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-base all
2017.20180305-1 [951 kB]
Get:43 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-
recommended all 2017.20180305-1 [14.9 MB]
Get:44 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-pictures
all 2017.20180305-1 [4,026 kB]
Get:45 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-latex-
extra all 2017.20180305-2 [10.6 MB]
Get:46 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tipa all 2:1.3-20
[2,978 kB]
Get:47 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-xetex all
2017.20180305-1 [10.7 MB]
Fetched 146 MB in 3s (58.2 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages ...
Selecting previously unselected package fonts-droid-fallback.
(Reading database ... 155047 files and directories currently installed.)
Preparing to unpack .../00-fonts-droid-fallback_1%3a6.0.1r16-1.1_all.deb ...
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1) ...
Selecting previously unselected package fonts-lato.
Preparing to unpack .../01-fonts-lato_2.0-2_all.deb ...
Unpacking fonts-lato (2.0-2) ...
Selecting previously unselected package poppler-data.
Preparing to unpack .../02-poppler-data_0.4.8-2_all.deb ...
Unpacking poppler-data (0.4.8-2) ...
Selecting previously unselected package tex-common.
Preparing to unpack .../03-tex-common_6.09_all.deb ...
Unpacking tex-common (6.09) ...
Selecting previously unselected package fonts-lmodern.
```

```
Preparing to unpack .../04-fonts-lmodern_2.004.5-3_all.deb ...
Unpacking fonts-lmodern (2.004.5-3) ...
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack .../05-fonts-noto-mono_20171026-2_all.deb ...
Unpacking fonts-noto-mono (20171026-2) ...
Selecting previously unselected package fonts-texgyre.
Preparing to unpack .../06-fonts-texgyre_20160520-1_all.deb ...
Unpacking fonts-texgyre (20160520-1) ...
Selecting previously unselected package javascript-common.
Preparing to unpack .../07-javascript-common_11_all.deb ...
Unpacking javascript-common (11) ...
Selecting previously unselected package libcupsfilters1:amd64.
Preparing to unpack .../08-libcupsfilters1_1.20.2-0ubuntu3.1_amd64.deb ...
Unpacking libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) ...
Selecting previously unselected package libcupsimage2:amd64.
Preparing to unpack .../09-libcupsimage2_2.2.7-1ubuntu2.8_amd64.deb ...
Unpacking libcupsimage2:amd64 (2.2.7-1ubuntu2.8) ...
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack .../10-libijs-0.35_0.35-13_amd64.deb ...
Unpacking libijs-0.35:amd64 (0.35-13) ...
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack .../11-libjbig2dec0_0.13-6_amd64.deb ...
Unpacking libjbig2dec0:amd64 (0.13-6) ...
Selecting previously unselected package libgs9-common.
Preparing to unpack .../12-libgs9-common_9.26~dfsg+0-0ubuntu0.18.04.14_all.deb
...
Unpacking libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.14) ...
Selecting previously unselected package libgs9:amd64.
Preparing to unpack .../13-libgs9_9.26~dfsg+0-0ubuntu0.18.04.14_amd64.deb ...
Unpacking libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.14) ...
Selecting previously unselected package libjs-jquery.
Preparing to unpack .../14-libjs-jquery_3.2.1-1_all.deb ...
Unpacking libjs-jquery (3.2.1-1) ...
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack .../15-libkpathsea6_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libpotrace0.
Preparing to unpack .../16-libpotrace0_1.14-2_amd64.deb ...
Unpacking libpotrace0 (1.14-2) ...
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack .../17-libptexenc1_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package rubygems-integration.
Preparing to unpack .../18-rubygems-integration_1.11_all.deb ...
Unpacking rubygems-integration (1.11) ...
Selecting previously unselected package ruby2.5.
```

```
Preparing to unpack .../19-ruby2.5_2.5.1-1ubuntu1.10_amd64.deb ...
Unpacking ruby2.5 (2.5.1-1ubuntu1.10) ...
Selecting previously unselected package ruby.
Preparing to unpack .../20-ruby_1%3a2.5.1_amd64.deb ...
Unpacking ruby (1:2.5.1) ...
Selecting previously unselected package rake.
Preparing to unpack .../21-rake_12.3.1-1ubuntu0.1_all.deb ...
Unpacking rake (12.3.1-1ubuntu0.1) ...
Selecting previously unselected package ruby-did-you-mean.
Preparing to unpack .../22-ruby-did-you-mean_1.2.0-2_all.deb ...
Unpacking ruby-did-you-mean (1.2.0-2) ...
Selecting previously unselected package ruby-minitest.
Preparing to unpack .../23-ruby-minitest_5.10.3-1_all.deb ...
Unpacking ruby-minitest (5.10.3-1) ...
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack .../24-ruby-net-telnet_0.1.1-2_all.deb ...
Unpacking ruby-net-telnet (0.1.1-2) ...
Selecting previously unselected package ruby-power-assert.
Preparing to unpack .../25-ruby-power-assert_0.3.0-1_all.deb ...
Unpacking ruby-power-assert (0.3.0-1) ...
Selecting previously unselected package ruby-test-unit.
Preparing to unpack .../26-ruby-test-unit_3.2.5-1_all.deb ...
Unpacking ruby-test-unit (3.2.5-1) ...
Selecting previously unselected package libruby2.5:amd64.
Preparing to unpack .../27-libruby2.5_2.5.1-1ubuntu1.10_amd64.deb ...
Unpacking libruby2.5:amd64 (2.5.1-1ubuntu1.10) ...
Selecting previously unselected package libsynctex1:amd64.
Preparing to unpack .../28-libsynctex1_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libtexlua52:amd64.
Preparing to unpack .../29-libtexlua52_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
.../30-libtexluajit2_2017.20170613.44572-8ubuntu0.1_amd64.deb ...
Unpacking libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack .../31-libzzip-0-13_0.13.62-3.1ubuntu0.18.04.1_amd64.deb ...
Unpacking libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) ...
Selecting previously unselected package lmodern.
Preparing to unpack .../32-lmodern_2.004.5-3_all.deb ...
Unpacking lmodern (2.004.5-3) ...
Selecting previously unselected package preview-latex-style.
Preparing to unpack .../33-preview-latex-style_11.91-1ubuntu1_all.deb ...
Unpacking preview-latex-style (11.91-1ubuntu1) ...
Selecting previously unselected package t1utils.
```

```
Preparing to unpack .../34-t1utils_1.41-2_amd64.deb ...
Unpacking t1utils (1.41-2) ...
Selecting previously unselected package tex-gyre.
Preparing to unpack .../35-tex-gyre_20160520-1_all.deb ...
Unpacking tex-gyre (20160520-1) ...
Selecting previously unselected package texlive-binaries.
Preparing to unpack .../36-texlive-
binaries_2017.20170613.44572-8ubuntu0.1_amd64.deb ...
Unpacking texlive-binaries (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package texlive-base.
Preparing to unpack .../37-texlive-base_2017.20180305-1_all.deb ...
Unpacking texlive-base (2017.20180305-1) ...
```

14 Alternative way to convert pdf

If the above method does not work, please look into this instruction.