The key concepts in our Problem Set:

1. Although L_2 regularization is the same as doing weight decay for Stochastic Gradient Descent (SGD), it is not for adaptive optimization methods such as Adam. This is further confused by Pytorch's choice of argument names for its Adam and AdamW implementations. AdamW, by definition, is a weight-decoupled adaptive stochastic optimization method that implements weight decaying effects onto Adam (ADAptive Moment estimation) through entering in corresponding values for the *weight_decay* argument. But if you take a look at PyTorch's documentation on Adam, we also see a *weight_decay* parameter for us to set - which makes one wonder, "setting this parameter for Adam should be equivalent to doing the same with AdamW - right?". This *weight_decay* parameter actually is used to change the L_2 regularization in Adam, with the default being no regularization. In this problem, we will explore / prove how these two methods are not equivalent and how *weight_decay* in Adam is a potential misnomer as L_2 regularization does not equal weight decay in Adam. Along the way, we also introduce the ideas of weight decoupling, fundamentals of adaptive gradient methods, and its relationship with L_2 regularization and weight decay. Finally, we will convince you that AdamW, a weight decoupled version of Adam, leads to a decoupled hyperparameter space that is better behaved and easier to tune.
2. The first thing the paper (Source 1) covers is that L_2 regularization implies weight decay for SGD. We covered this in lecture to show the equivalence. In Q 1.a.i, we have the student start with showing this equivalence as a warm up. They should find that the implied weight decay factor is equal to the L_2 penalization term scaled by the learning rate of the method.
3. When the weight decay factor and learning rate depend on each other, we say our method is coupled. We are constraining our search in hyperparameter space as one is coupled with the other. We can decouple these relationships of hyperparameters on both SGD and Adam through a simple modification of each of the algorithms.
    a. In part 1.a.ii, we explain the idea of decoupling covered in the paper (Source 1) as so: We introduce the idea of *weight decoupling* - a technique to decouple the relationship between the weight decay factor and learning rate in our code implementation. The weight decoupled SGD algorithm with weight decay is known as SGDW. SGDW allows us to search through both hyperparameter spaces independently. We then have the student decouple SGD with momentum, creating SGDW with momentum. This problem can be used to give intuition to students when deriving AdamW (implicit problem in Q 2.a).
4. In Q 1.b we have the student try to prove that L_2 regularization does not give the same effect as weight decay under Adaptive Gradient Methods such as AdaGrad and RMSProp shown in the paper (Source 1). Unfortunately, Adam cannot fall under this proof entirely due to the algorithm's additional complexity in containing momentum and using the bias-corrected momentum term for weight update. This was something that was not explicitly mentioned in the paper (Source 1). But Adam, containing adaptive methods in its core structure, can get very close to a convincing argument that weight decay does not equal L_2 regularization.
    a. To get as convincing of an argument as possible to show this relationship for

Adam, we introduce RMSProp with a bias-correcting second moment. This method is made for the sake of constructing an algorithm as close to Adam as possible while still being able to fit into the proof. In this problem, we give the pseudocode of the method and have students solve for the M_t matrix for adaptive gradient update in terms of the outer product of subgradients (G_t) and the second moment factor ($beta_2$). The reason why we are deriving M_t in terms of G_t is because this term is the heart component of adaptive gradient methods.

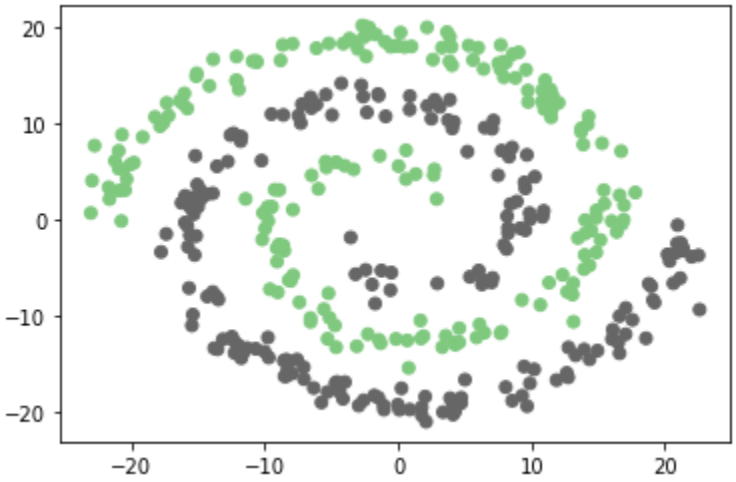
   i. **Therefore, once we prove that RMSProp with bias-correcting second moment falls under the proof of inequivalence between L_2 regularization and weight decay, here are couple of things we can argue to extend this proof to Adam the best of our abilities:**
      1. The momentum term that Adam has in addition to RMSProp with bias-correcting second moment shouldn't affect the inequivalence between L_2 regularization and weight decay. We argue this because as we saw in part 1.a.ii, having a momentum term in SGD did not affect the equivalence of L_2 regularization and weight decay.
      2. The bias-correcting term for momentum can also use a similar argument in that, since we have a bias-correcting term for the second moment that is encapsulated in RMSProp with bias-correcting second moment, it shouldn't affect the inequivalence of L_2 regularization and weight decay.

b. After students drive the M_t matrix, we show that if M_t is not a multiple of identity, which is generally the case to take advantage of the adaptive nature of the optimization, there exists no L_2 coefficient such that running adaptive gradient methods through the loss function with L_2 regularization would be equivalent to weight decay.

c. Lastly, the paper (Source 1) makes an interesting claim on the instances where we can scale-adjust the L_2 term to have some regularization that would induce weight decaying effects. We extend this idea to construct an iteration-dependent loss function with varying regularization term to show that we can implement a loss function that would induce weight decaying effects for adaptive gradient methods. This isn't something that is necessarily practical / should be used but rather an interesting way to encapsulate weight decaying effects into L_2 regularization terms even in adaptive gradient methods.

5. Decoupling Adam leads to AdamW which (although is still a topic of research) substantially improves Adam's generalization performance allowing it to compete with SGD with momentum
   a. To introduce the students to the Adam and AdamW algorithms in Q 2.a we have them code the two algorithms (fill in blank lines in code we made) in python. The methods they fill in should match the functionality of Pytorch's optimizers. We included an autograder by running their optimizers in a deterministic way that should output the same loss as ours so the student can sanity check that they understand the code.

b. We created a synthetic dataset to train a relatively simple feed forward network of 3 hidden layers (we encourage students to experiment with different model architectures) in order to perform a classification task. The dataset is shown below. The complexity of the dataset is also customizable by adjusting the amount of noise, number of data points, and the number of times the two colors spiral around each other.

1. 

c. In the paper, there are some nice visualizations demonstrating the optimal value ranges for learning rate and weight decay for Adam (weight decay would be the L_2 for Adam) and AdamW. We run training simulations over a large combination of (learning_rate, weight_decay) values (default search is over a 16x24 grid), and record the final losses and accuracies of the model to plot in a heatmap. We have the students run code to make our recreation of these visualizations. The main pattern they should see is that the parameters are decoupled for AdamW and there is a larger range of parameters to get the optimal loss since AdamW tends to generalize better than Adam with L_2. To make sure the student sees this we ask them a few questions:
    i. What similarities and differences do you notice between the accuracy heat maps of the two algorithms? What about similarities and differences between the loss heat maps of the two algorithms?
    ii. Adjust the difficulty of the data in some way, either by making it “harder” or “easier” (see notebook for a more detailed description), and report on differences between heat maps generated for the two algorithms and also differences between heat maps from part a.
    iii. What can you conjecture about the processes of tuning hyperparameters of L_2 regularized Adam and AdamW? Are there differences in the absolute best validation accuracy of the two algorithms? What about differences in the frequency of “good” hyperparameter settings above a certain threshold (say 90%) between the two algorithms?

Sources:

1. Main paper - Loshchilov, Ilya, and Frank Hutter. 'Decoupled Weight Decay Regularization'. 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. https://arxiv.org/pdf/1711.05101.pdf

(Additional sources not cited but used in the project found in LaTeX pdf)

Wish list:

1. The original paper states that Adam can substantially benefit from a scheduled learning rate multiplier. The paper shows that significant improvements in results come from implementation of cosine annealing and warm restarts. This is something we wish we had the time to implement and play around with.
2. We also would have liked to test and visualize the differences between the two algorithms on a non-synthetic dataset such as MNIST or CIFAR-10 (this is what the original authors experimented on). However, this would have taken too much computational resources to perform as exhaustive of a hyperparameter search as we did.
3. There were a couple of interesting sources we've found that talk about all the small justifications around Adam optimizers (bias-correcting and why we need them) and AdaGrad with very interesting visualizations (e.g. rolling a ball over to find the optimal point - comparing momentum, SGD, AdaGrad, Adam, etc.). If we had the time, we can maybe construct a problem that dives into more of the fundamentals of these optimizers with visualizations that can really assist students' intuition. But given that we want to stick more to what our main paper talked about, we decided to not go too deep into this for this final project / problem set. Nevertheless, this would be something that we would've loved to see potentially incorporated in the class to introduce students in the future.
4. More notes on connecting Adam and AdamW on practical State of the Art neural network architectures like transformers. We've attached an interesting survey of looking at BERT and assessing various Adam optimizers. Attaching and making some note of these recent researches on optimizers can potentially make students more motivated and interested in learning more about optimizers' capabilities and its surrounding intuition on when and what to use.