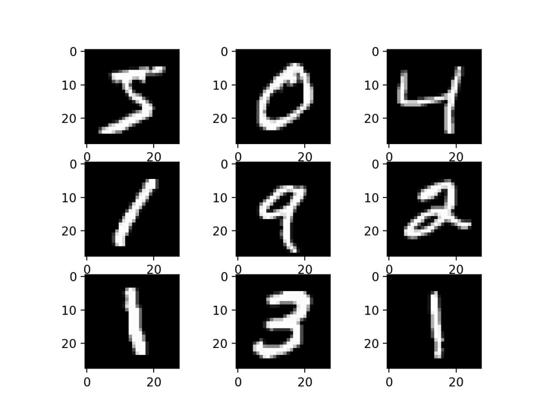
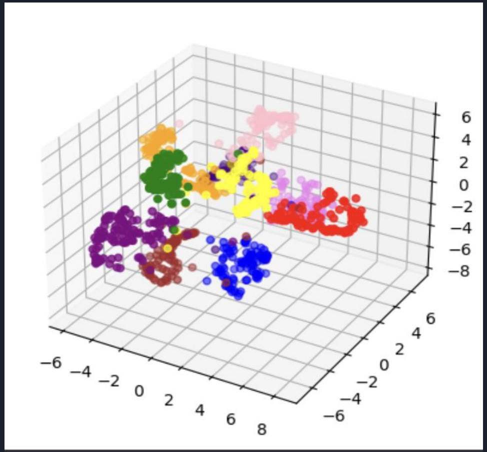
Explorations in Latent Space

Michael Murphy

Goals

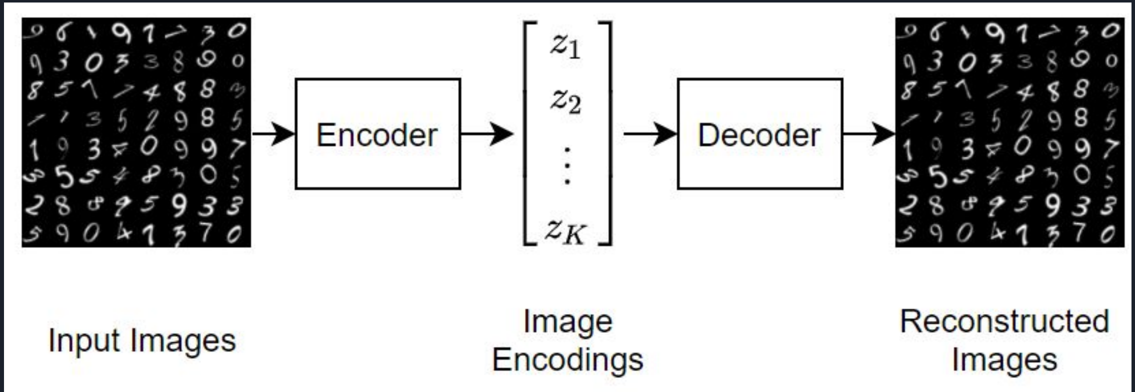
- Last Summer I learned PCA w/ Libby
- This summer my goals were:
    - Reimplement PCA as a linear AE
    - Improve performance by constructing nonlinear AE
    - Become familiar with Jax and related libraries
    - Explore the question: what factors influence the structure of a representation?

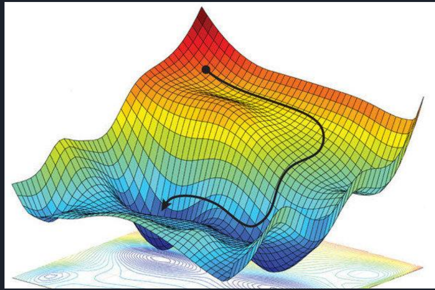




Linear Autoencoder

1. Transform 728d image vector (28x28) into 10d subspace (encode) and then back into 728d (decode) via matrix multiplication
2. Evaluate accuracy by comparing reconstructed image to original (MSE Loss Objective Function)
3. Optimize parameters (initialized as random matrices) by performing gradient descent on loss function

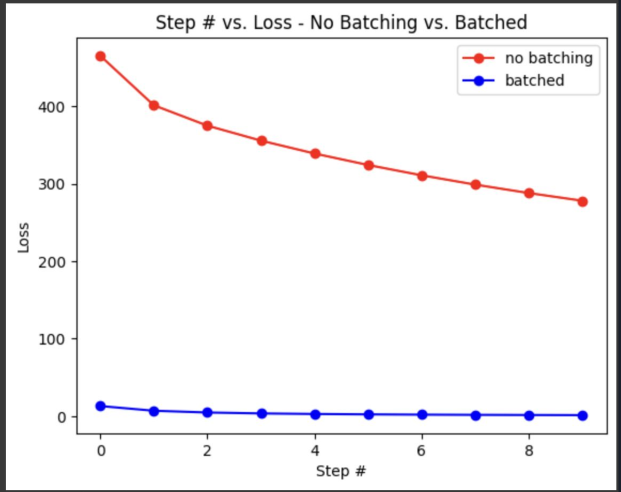




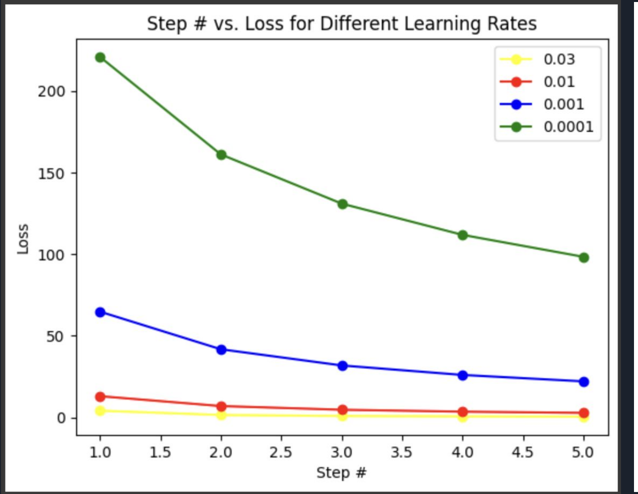
$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

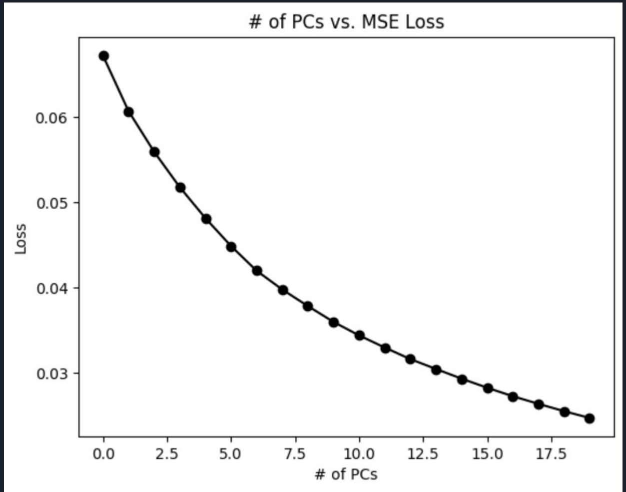
What Affects Performance of Linear AE?

- Number of steps
- Batching (Stochastic Gradient Descent)
- Learning rate
- # of Latent Dimensions



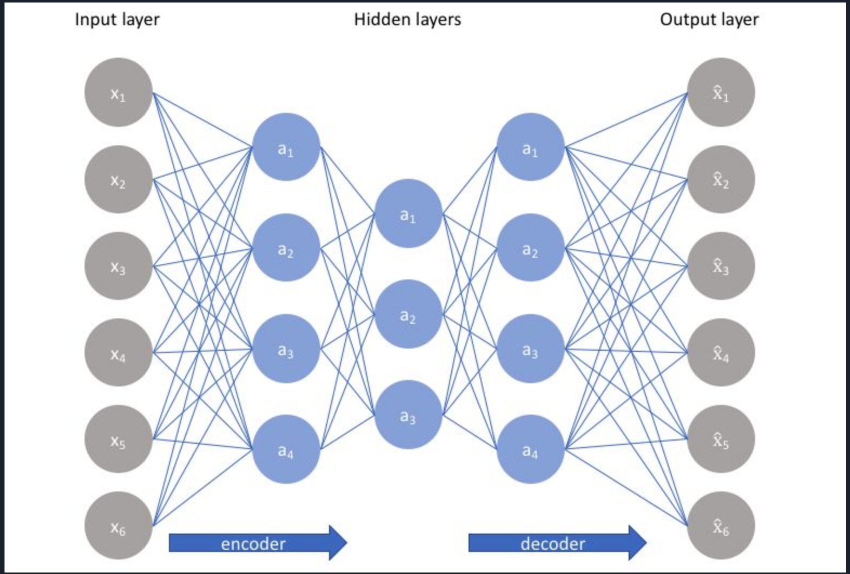
No Batching Time: 14.78401s, Batching Time: 12.97605s

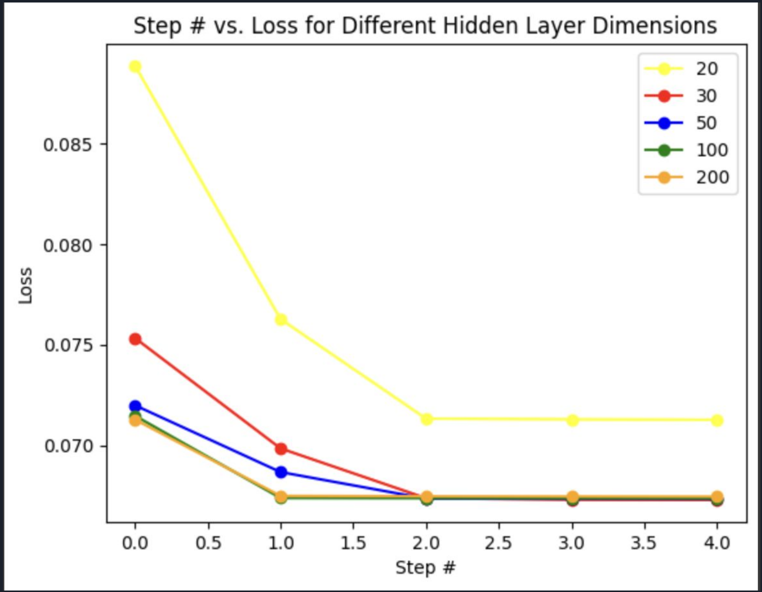




Nonlinear AE

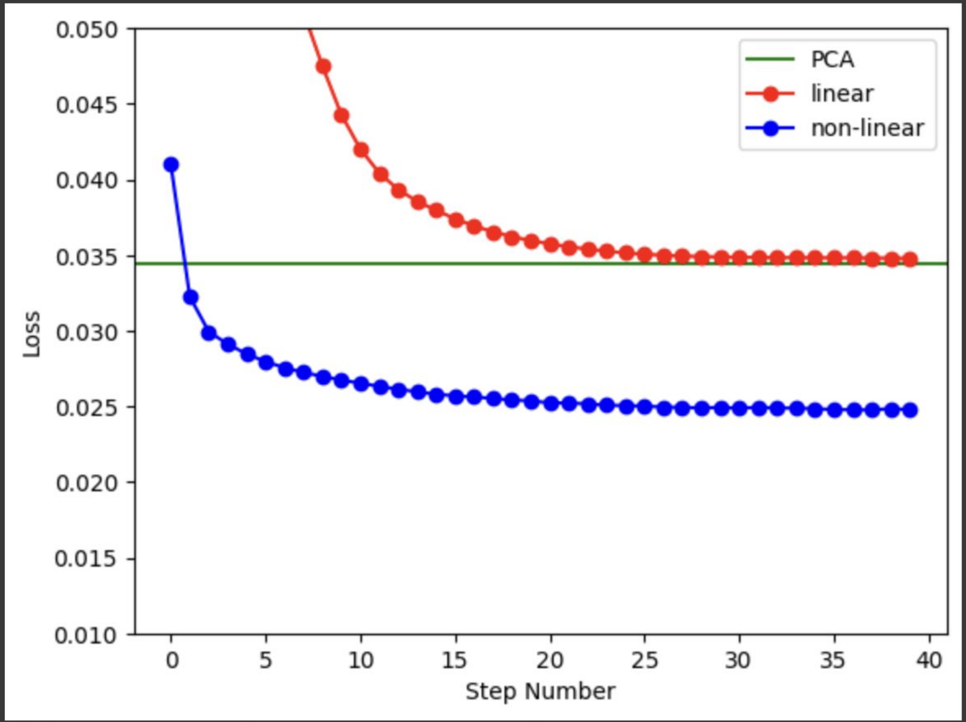
- Same idea as linear AE, but contains “hidden layers” that perform nonlinear transformations (relu, tanh)
- Can capture more complex patterns in the data
- Number and size of hidden layer(s) affects performance





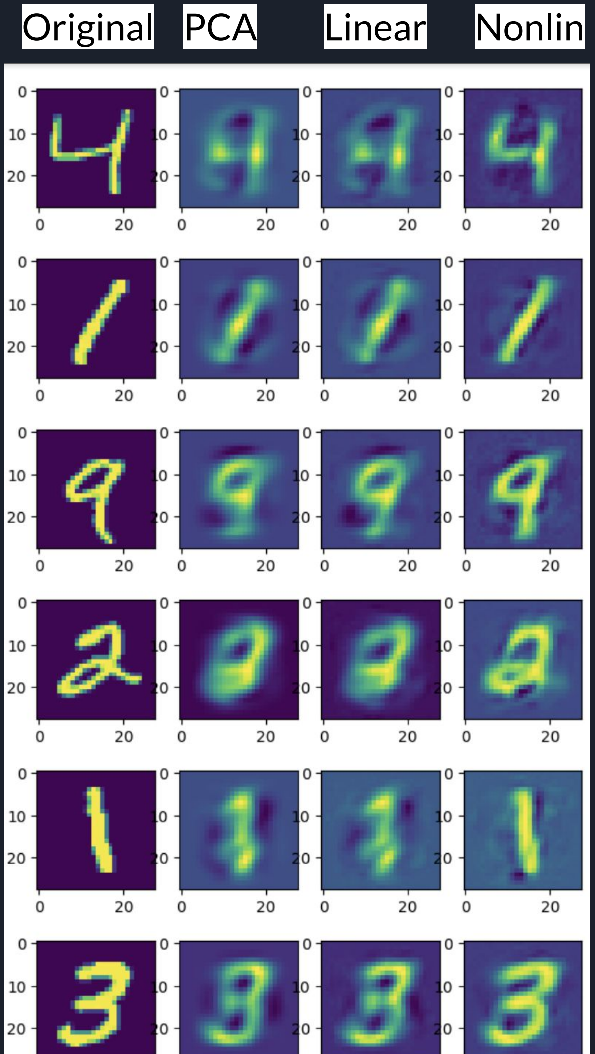
Linear AE vs. Nonlinear AE vs. PCA Loss

- PCA performance is equivalent to the best possible linear AE



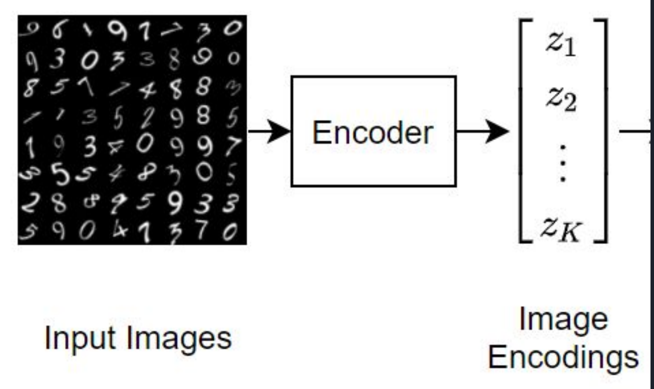
Minimum Losses: Linear = 0.03479, Nonlinear = 0.02478, PCA = 0.03442

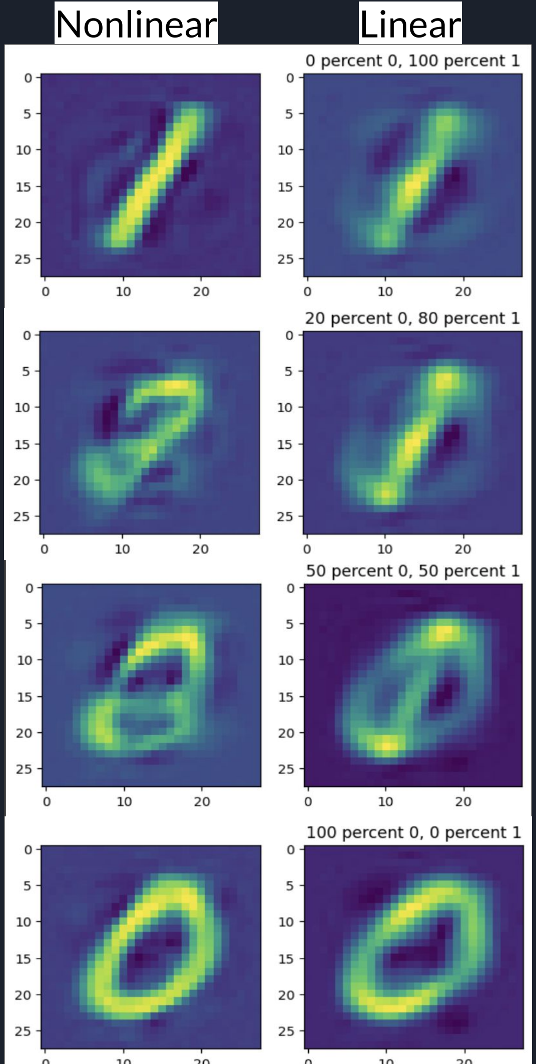
Image Reconstruction



Interpolation

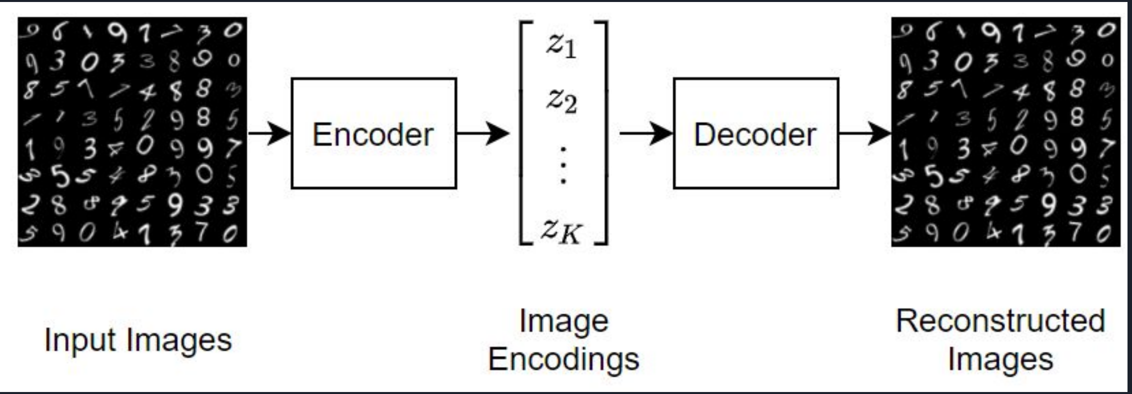
- Can we decode a combination of the z vectors for 2 different digits to yield an image that is a mix of the 2 digits?
- Linear AE just crossfades between the 2 digits
- Nonlinear AE decodes interesting digit-like images

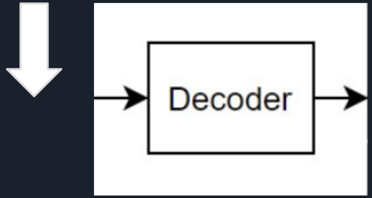




Supervised AE

- Can the model learn to reconstruct the image AND classify its digit from a lower dimensional representation?
- New objective function to optimize (once again with SGD): Reconstruction Loss AND Classification Loss

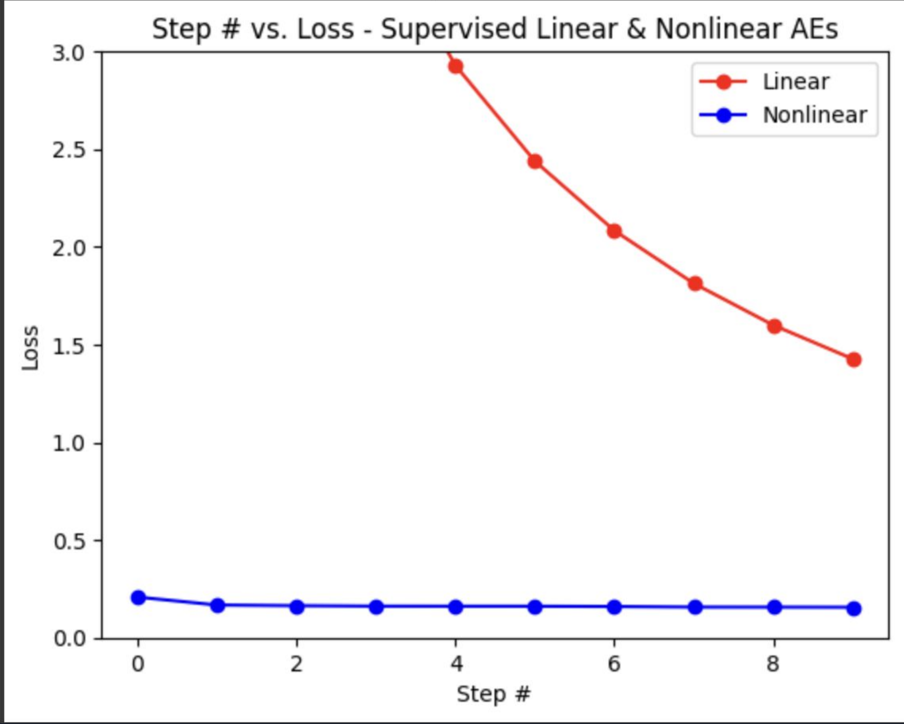




Digit Classification (i.e. 3, 7)

Linear vs. Nonlinear Supervised AE

```
Minimum Losses: Linear = 1.42626, Nonlinear = 0.15593
```



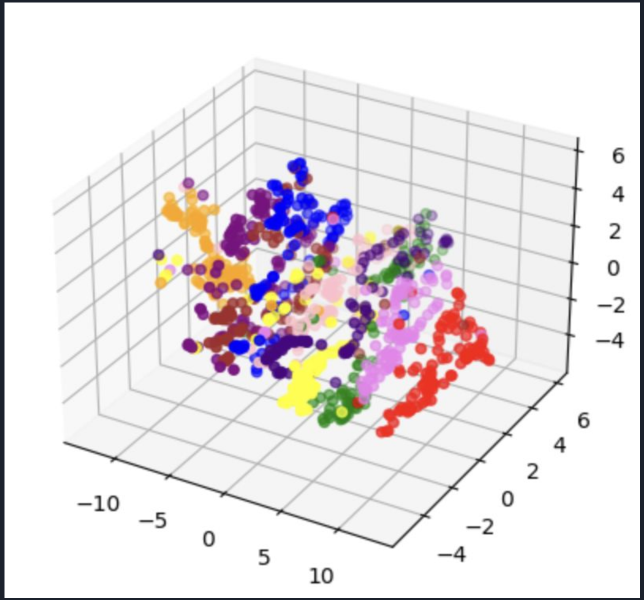

```
def accuracy(data, labs, params, end):
  correct = 0.
  for i in range(end):
    z = data[i] @ params[0] + params[1]
    y_hat = z @ params[4] + params[5]

    pred = jnp.where(y_hat == max(y_hat))[0][0]
    real = jnp.where(labs[i] == max(labs[i]))[0][0]

    if (pred == real):
      correct += 1

  return (correct / end)
```

```
Linear AE Accuracy (train data): 0.81
Nonlinear AE Accuracy (train data): 0.961
Linear AE Accuracy (test data): 0.802
Nonlinear AE Accuracy (test data): 0.947
```

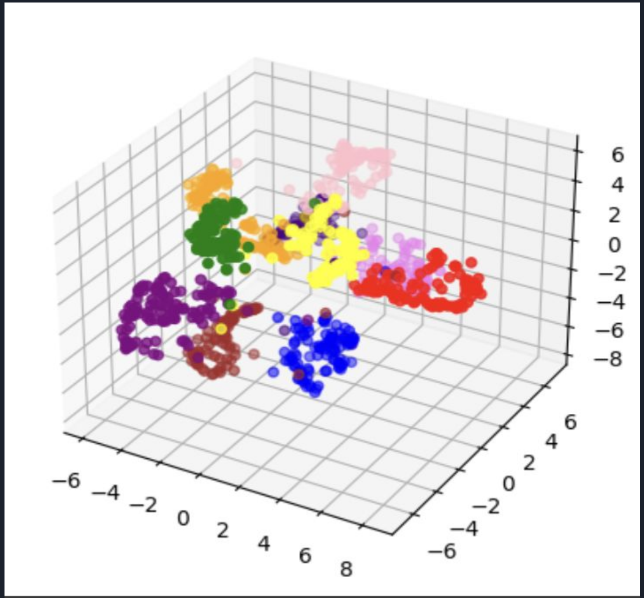
Supervised AE vs Unsupervised AE

- Used T-SNE to visualize the 10D representations of each digit in 3d
- Supervision makes digits more distinguishable!
- Switch to 3d interactive plots

Unsupervised



Supervised



Jax

Main Takeaways:

- Functional Programming
- grad()
- vmap()
- Just In Time (JIT) Compilation

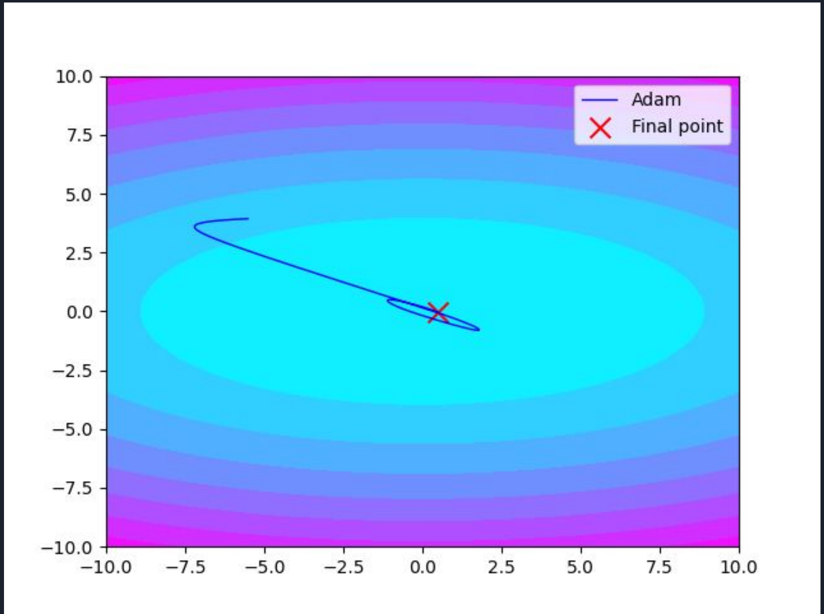
Timing jit vs. non-jit training functions

```
#@title Timing jit vs. non-jit training functions

# timing for non jit training function - 10 steps
t0 = time.time()
for i in range(10):
  theta = train_step(x,theta,0.0001)
t1 = time.time()
print(f"non-jit time: {t1 - t0}")

# timing for jit train function - 10 steps
t0 = time.time()
for i in range(10):
  theta = train_step_jit(x,theta,0.0001)
t1 = time.time()
print(f"jit time: {t1 - t0}")
```

```
non-jit time: 20.077202320098877
jit time: 6.219285488128662
```

Optax

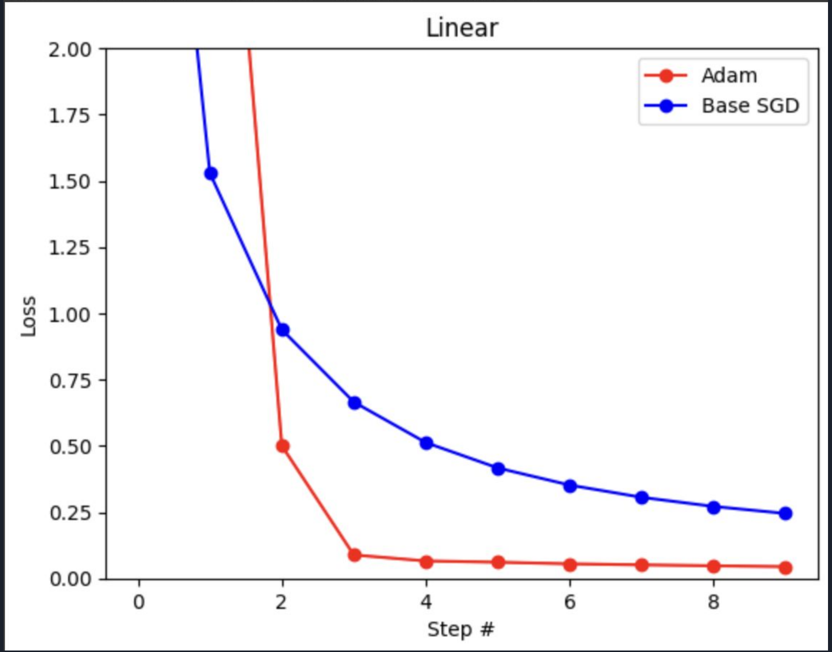
- Optimization Library for Jax
- Adam improves over SGD by scaling the gradient in terms of the curvature (momentum)
    - Helps with tuning the learning rate



Optax

- Adam is slower (usually) for a given number of steps, but significantly more accurate!

```
Minimum Losses: Adam = 0.04428, Base SGD = 0.24485
Run Times: Adam = 12.904s, Base SGD = 10.997s
```



Killifish Data

- Last year, Libby and I performed PCA on 1.2mil x 12 array of Killifish motion data
- How would my AEs perform?

	frame_count	frame_timestamp	x_snout	y_snout	x_midbody	y_midbody	x_sidebody	y_sidebody	x_endbody	y_endbody	x_tail	y_tail	x_fan	y_fan
0	0	1.619461e+09	109.540108	142.976059	124.138733	135.067261	NaN	NaN	151.504974	123.650360	165.433395	113.957596	180.608887	105.863663
1	1	1.619461e+09	107.265854	142.104813	121.822556	136.475708	NaN	NaN	147.926788	123.891907	160.844559	115.370964	174.711700	106.121666
2	2	1.619461e+09	105.351967	142.368988	119.697525	136.510712	NaN	NaN	145.834869	123.651062	160.248383	116.183006	174.869812	107.406532
3	3	1.619461e+09	103.469025	142.543152	117.734238	136.160843	NaN	NaN	143.201263	124.122238	158.663406	117.667580	176.247086	109.998772
4	4	1.619461e+09	102.180344	142.780396	115.806297	136.204010	NaN	NaN	142.057327	123.676147	157.429688	117.932121	176.791046	110.939461
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1723651	1726808	1.619548e+09	84.945068	53.992599	96.640488	65.568367	NaN	NaN	108.316635	89.087357	109.255569	107.692741	109.788170	128.898438
1723652	1726809	1.619548e+09	83.437828	49.815742	97.773056	63.414818	NaN	NaN	112.321381	86.499474	113.034012	104.349236	106.420898	123.391586
1723653	1726810	1.619548e+09	82.824081	46.829994	97.266914	60.762054	NaN	NaN	114.773003	81.535423	120.126129	98.978592	110.384720	118.568207
1723654	1726811	1.619548e+09	81.770180	42.896610	94.724854	55.871910	NaN	NaN	115.567490	77.666855	122.965393	94.156998	117.896706	115.093361
1723655	1726812	1.619548e+09	81.606567	42.340202	93.756645	53.572468	NaN	NaN	114.121025	74.155083	124.118027	88.861328	125.768005	111.091743

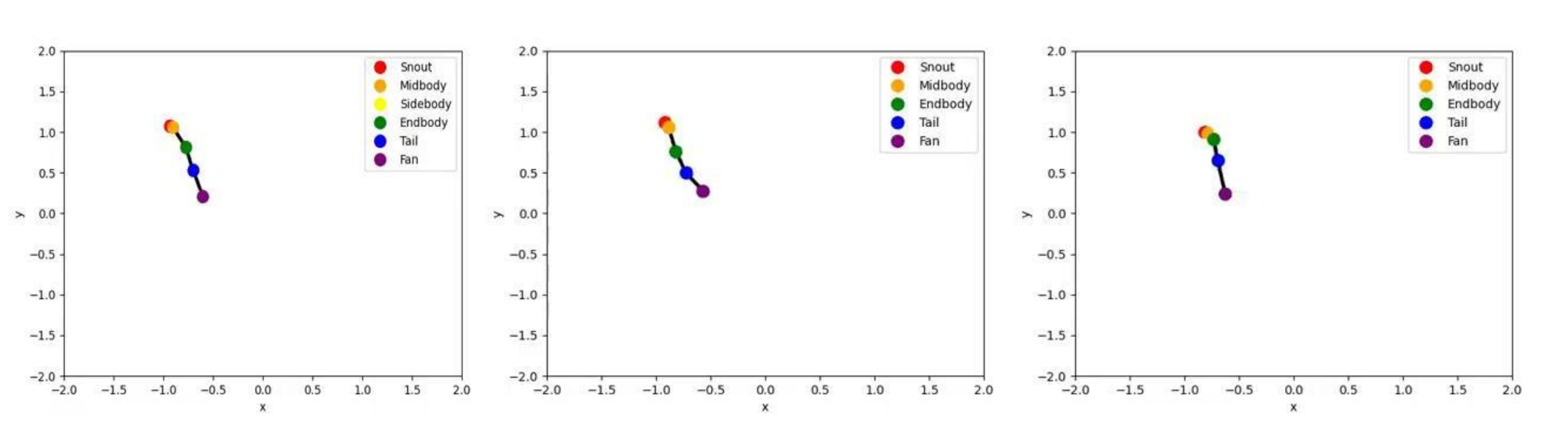
Killifish Data

- Animations of the killifish motion for the first 1000 frames
- 12d data mapped to 4d and then reconstructed
- Linear AE reconstruction captures only the general position
- Nonlinear AE can capture more complex movement (tail flapping)

Original

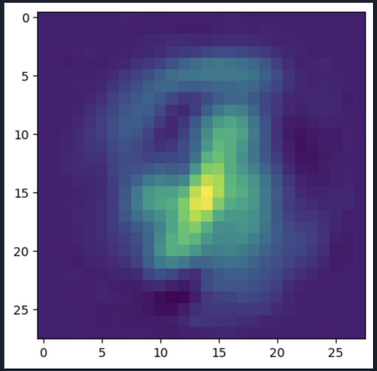
Linear

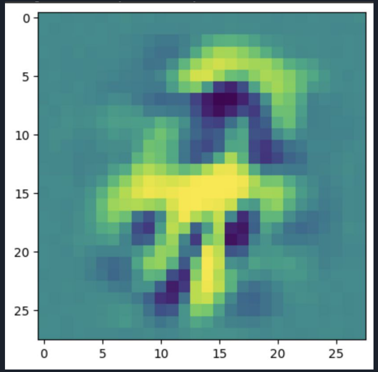
Nonlinear



Future Plans

- Hawaii
- Berkeley Statistics
    - DATA 8
    - Multivariable Calc
    - Linear Algebra
- Variational AE for Image Generation





Thank You So Much To...

- Scott
- Yixiu
- Linderman Lab
- Stack overflow