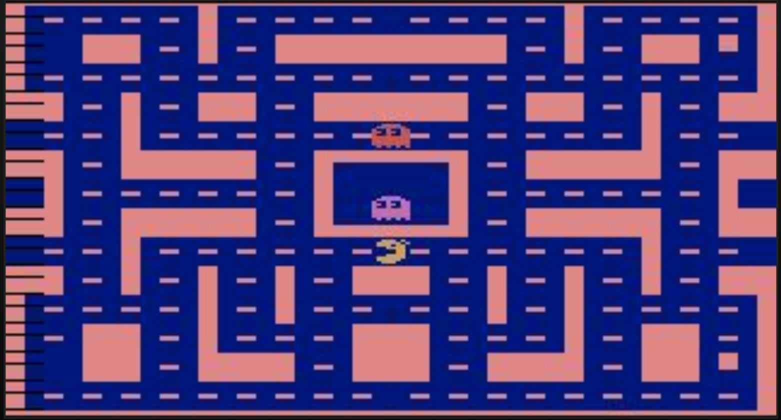
Ms. Pacman (Atari)

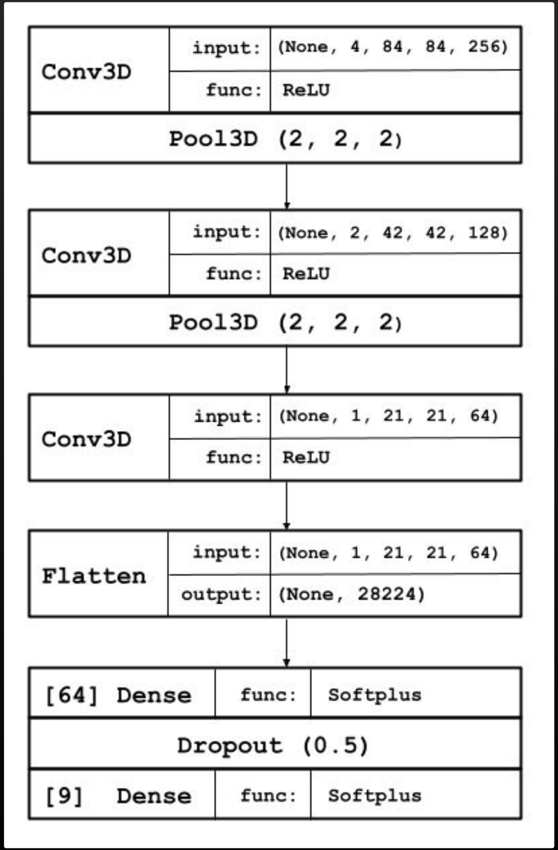
Image Processing Model



Kaden Empey

Model

```
# stack_count => 4, frame_size => 84, channel_count => 1

kernel_size = (stack_count, 4, 4)
pool_size = (2, 2, 2)
input_shape = (stack_count, frame_size, frame_size, channel_count)

model = keras.models.Sequential()
model.add(keras.layers.Conv3D(256, kernel_size=kernel_size, activation="relu",
padding="same", input_shape=input_shape))
model.add(keras.layers.MaxPool3D(pool_size=pool_size))

model.add(keras.layers.Conv3D(128, kernel_size=kernel_size, activation="relu",
padding="same"))
model.add(keras.layers.MaxPool3D(pool_size=pool_size))

model.add(keras.layers.Conv3D(64, kernel_size=kernel_size, activation="relu",
padding="same"))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(64, activation="softplus"))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(action_count, activation="softplus"))

model.compile(loss="mse", optimizer="nadam", metrics=["mae"])
```



Conv3D
input: (None, 4, 84, 84, 256)
func: ReLU
Pool3D (2, 2, 2)

Conv3D
input: (None, 2, 42, 42, 128)
func: ReLU
Pool3D (2, 2, 2)

Conv3D
input: (None, 1, 21, 21, 64)
func: ReLU

Flatten
input: (None, 1, 21, 21, 64)
output: (None, 28224)

[64] Dense
func: Softplus
Dropout (0.5)
[9] Dense
func: Softplus

Parameters

Γ → 0.99-0.995

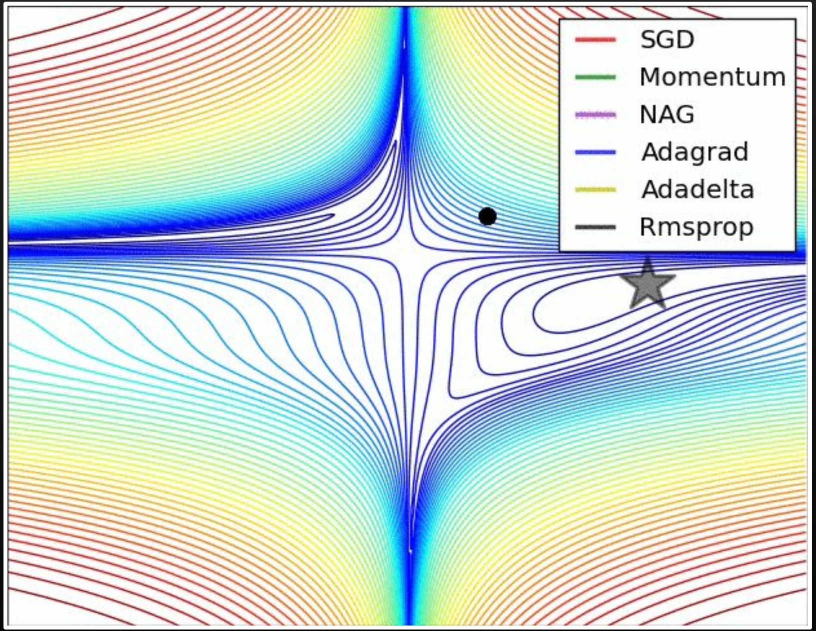
Ε → 0.40-0.01

Reward

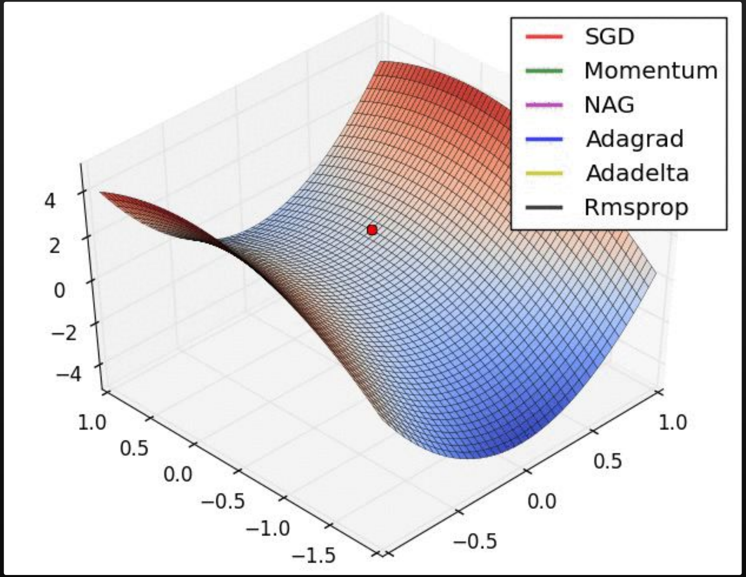
env.step() 5x

+1 per frame

GD Optimization Algorithms



https://jlmelville.github.io/mize/nesterov.html

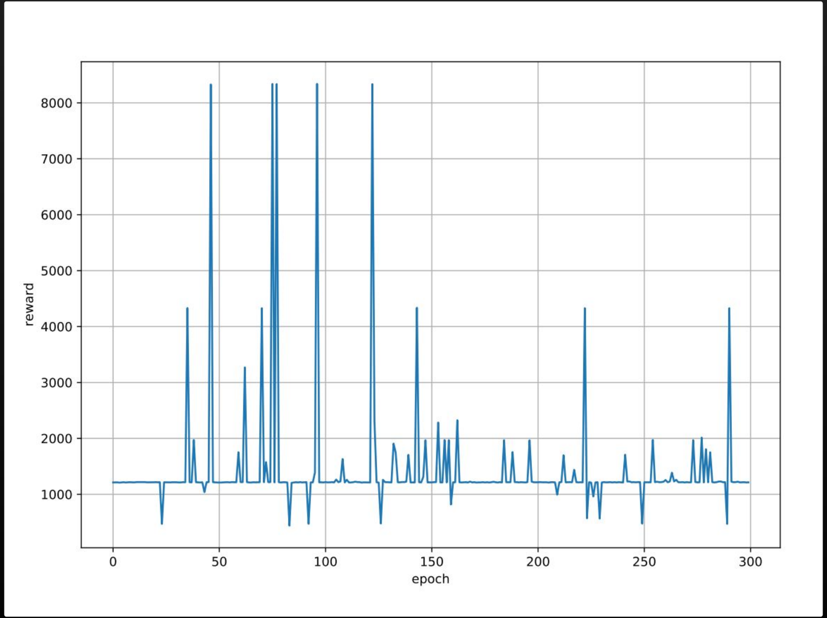


https://ruder.io/optimizing-gradient-descent/

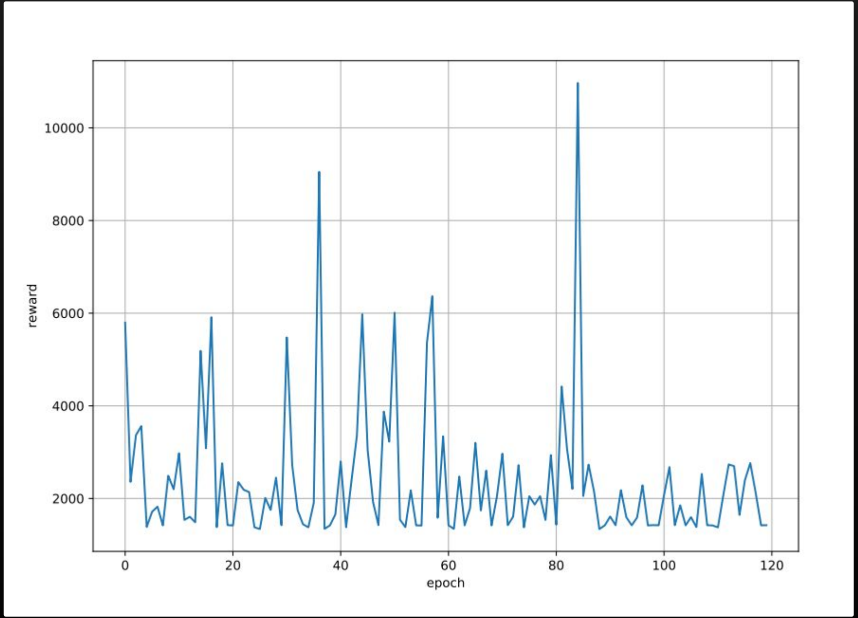
http://louistiao.me/notes/visualizing-and-animating-optimization-algorithms-with-matplotlib/

Adam performance

TERMINAL ON LIFE LOSS TRUE

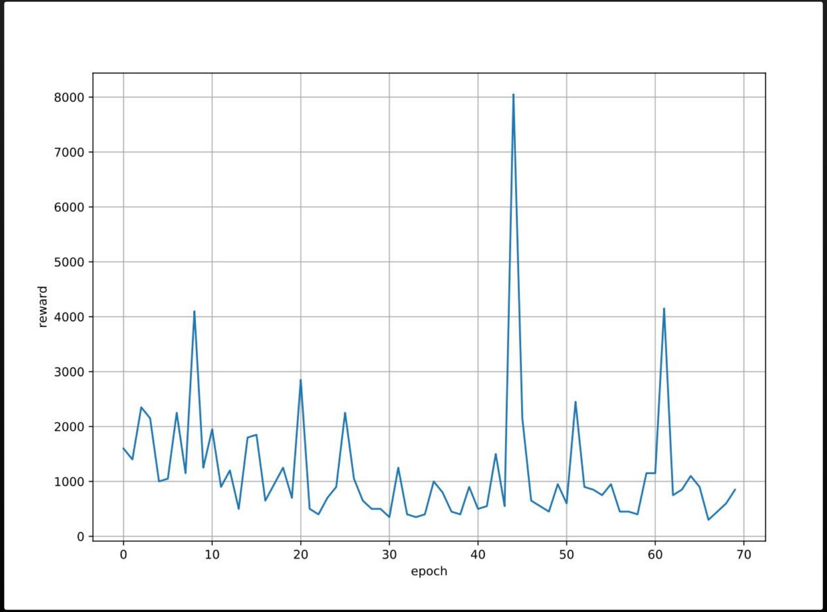


TERMINAL ON LIFE LOSS FALSE



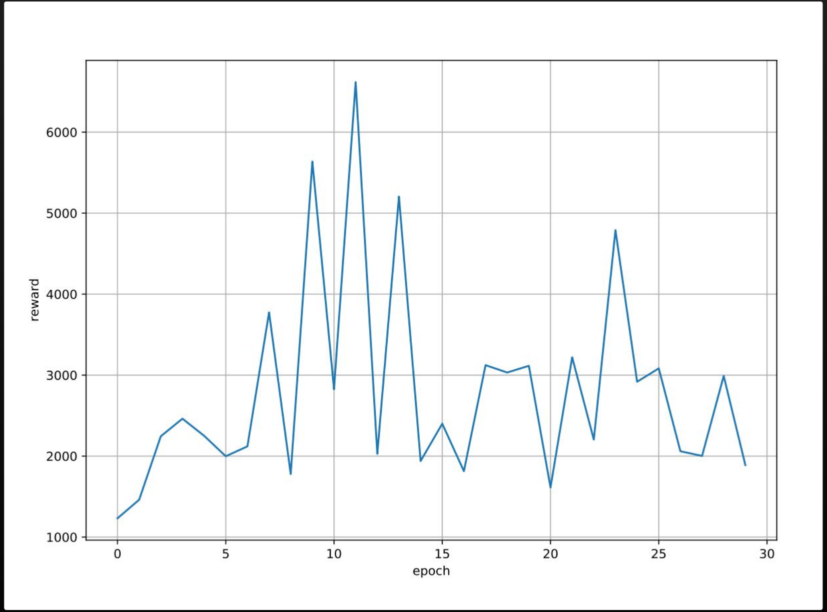
Nadam performance

TERMINAL ON LIFE LOSS TRUE



*no reward per frame

TERMINAL ON LIFE LOSS FALSE



Future Experimentation

- More learning from human input
- Different reward/penalty characteristics
  - Incentivize unique movements
  - Penalize periods of no reward
- Tweak Model
  - Assess different optimizers
  - Use kernel and bias initializers (e.g. he_normal, glorot_uniform..)
  - Try different loss functions (e.g. categorical_crossentropy, squared_hinge..)
- Game state as input (RAM)