GRU (Gated Recurrent Unit) models can be trained using various optimizers, depending on your specific use case and preferences. While Adam optimizer is commonly used for its good performance in a wide range of scenarios, you can experiment with other optimizers to see if they provide better results for your particular task. Here are a few alternatives:

1. Stochastic Gradient Descent (SGD): This is a classic optimizer that updates the weights based on the gradient of the loss with respect to the weights. It is computationally less expensive compared to more sophisticated optimizers like Adam.

python

* from tensorflow.keras.optimizers import SGD

optimizer = SGD(learning\_rate=0.01, momentum=0.9)

* RMSprop (Root Mean Square Propagation): RMSprop is an adaptive learning rate method that adjusts the learning rates of different parameters based on their historical gradients.

python

* from tensorflow.keras.optimizers import RMSprop

optimizer = RMSprop(learning\_rate=0.001, rho=0.9)

* Adagrad: Adagrad adapts the learning rates of the parameters based on the historical gradient information. It can be useful for sparse data.

python

* from tensorflow.keras.optimizers import Adagrad

optimizer = Adagrad(learning\_rate=0.01)

* Adadelta: Adadelta is an extension of Adagrad that aims to address its diminishing learning rates over time.

python

* from tensorflow.keras.optimizers import Adadelta

optimizer = Adadelta(learning\_rate=1.0, rho=0.95)

* Nadam: Nadam is an extension of Adam that incorporates the Nesterov accelerated gradient method.

python

1. from tensorflow.keras.optimizers import Nadam

optimizer = Nadam(learning\_rate=0.002, beta\_1=0.9, beta\_2=0.999)

When choosing an optimizer, it's essential to consider factors such as the nature of your data, the problem you're solving, and the characteristics of your model. Experimentation and tuning are often necessary to find the optimizer that works best for your specific scenario.