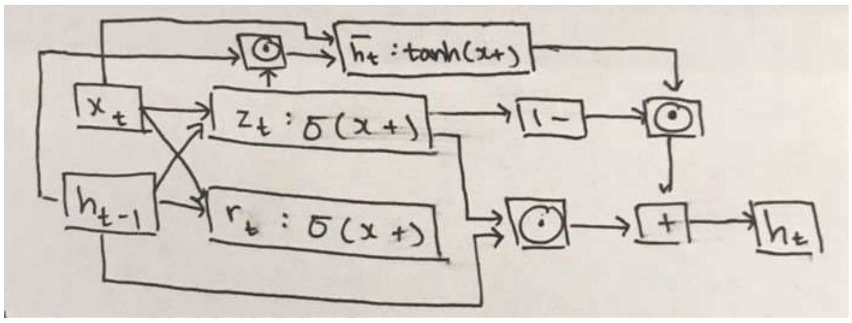
TASK 1

1.1:



1.2:
$W^z : d_h \times d_x$
$W^r : d_h \times d_x$
$W : d_h \times d_x$
$U^z : d_h \times d_h$
$U^r : d_h \times d_h$
$U^h : d_h \times d_h$

1.3:
The GRU operates using a reset gate and an update gate whereas the LSTM has an input, forget, update and output gate. For the GRU, the reset gate sits between the previous activation and the next cantidate activation to forget previous state, and the update gate decides how much of the candidate activation to use in updating the cell state. GRUs have fewer parameters and thus train a bit faster or need less data to generalize hence it is computationally more efficient due to the less complex structure. The GRU unit controls the information flow like the LSTM but without the memory unit. it just exposes the full hidden content without any control.

Task 2:

1.1: my way
Defensive distillation to prevent attacks that are based on gradients. The idea is to use softmax with a temperature T. Train a teacher network at T >> 1 and collect prediction probabilities p(x) for all the training samples x. Train a second network at the same large T >> 1 using p(x) as soft labels instead of 0/1 hard labels from the original dataset. At test time, use a new second network with T = 1. This would result in most predictions to have probabilities near 0/1, thus gradients are almost zero and numerically instable. This would cause attacks that are dependent on gradients to fail. Also, with a soft label provided to the second model. This uncertainty acts as an additional filter with an element of randomness to gaining a perfect match. This results in a far more robust algorithm that can spot spoofing attempts easier. Hence it adds flexibility to the algorithm's classification process so the model is less susceptible to exploitation. It creates a model whose surface is smoothed in the directions an adversary will typically try to explot, making it difficult for them to discover adversarial input tweaks that lead to misclassification.

1.2: second way
train the neural network with adverserial examples. Adverserial training: it is a brute force solution where we generate a lot of adverserial examples and explicitly train the model to not be fooled by each of them.

1.3: how attacker can overcome the first way (my way)
boundary attacks against black boxes can make defensive distillation fail. The idea is to move along decision boundary to the predicted class of the target image but alywas in the wrong way. use sample pertubations from a distribution to get a synth image that looks clsoe to target image but has wrong prediction. only accept if the perturbed image if it is adverserial. This approach has iteration of two steps to get close to the target image. This can prevent defensive distillation because methods that defend by hide/shatter gradients are ineffective against the boundary attack.

Task 3:

if the g_w is not capped. when we try to optimize the objective function, one can minimize the object by making g_w very negative and not bothering about the first term. Hence the goals encoded by the losses f_w would not be achieved.

if g_w is capped, it yields a better result as this means that one can not minimize the objective by making g_w very small. This will hence result in the goals encoded by the losses f_w and g_w to be achieved when we optimize this.

Task 4:

$$P(00) = 0.8 \times 0.8 = 0.64$$
$$P(01) = 0.8 \times 0.8 = 0.64$$
$$P(10) = 0.8 \times 0.8 = 0.64$$
$$P(11) = 0.8 \times 0.8 = 0.64$$