1.Our project idea is to build an AI that recognizes the alphabet and the numbers in sign language and that translates it back to english text. It uses a webcam to take pictures of signs.

2. The datasets we are working with are https://www.kaggle.com/datasets/grassknoted/asl-alphabet and https://www.kaggle.com/datasets/lexset/synthetic-asl-numbers. There are no changes to our initial choice of datasets because the datasets are huge and comprehensive. The datasets we are using are labeled, so we will be doing supervised learning. The test data in the datasets have high contrast and thus will be helpful to convert the targeted pixels into the required unit vectors. All the images in the datasest are of same dimensions, thus little preprocessing will be required, and we can easily convert all the images into our required dimensions. The first dataset contains alphabets, and the second one contains asl numbers, and we will implement our model to recognise these.
To improve our model, we will use data-augmentation techniques to create variations of the data we have in order to increase the size of our dataset and make sure our model can work in various situations.

3. a. For our model, we will be using the CNN classification algorithm. To implement it, we will use the PyTorch library. A CNN algorithm has normally 3 layers: a convolutional layer, a pooling layer, and a fully connected layer, which we will implement in our model (1). In the convolutional layer, we are going to use kernels to filter the images and extract useful data.
The pooling layer is used to decrease the size of the convolved feature (2). It reduces the size of the input and lead to smaller size of weights and less computation.We are thinking about using max pooling as our function for the pooling layer, since it is the most popular. We can alternate pooling layer and convolutional layer to reduce computing time.
For the fully connected-layer, we will use the Rectified Linear Unit (ReLU) function since it is one of the most popular right now. However, we will need to make sure we have a good learning rate, since this method is very fragile during training, because a large gradient can have such an impact that the neuron stop updating after that. If that doesn't work well, we can use a Softmax function. The fully connected-layer (or dense layer) is the final layer that outputs the probability that the image is in a certain class.
The other modules we will be using are numpy to vectorize the datasets and work with the vectors, and cv2.

https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

b. When training our model, we will use the Adam optimizer and our loss function if going to be cross-entropy. We will use a validation set to make sure our model is not overfitting.For regularization, we will use the dropout model. This regularization will be made after the layers

1:https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939#:~:text=A%20CNN%20typically%20has%20three,and%20a%20fully%20connected%20layer.
2:https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243
3: Regularization technique CNN and Keras:
https://medium.com/intelligentmachines/convolutional-neural-network-and-regularization-techniques-with-tensorflow-and-keras-5a09e6e65dc

with really high numbers of parameters, for example the fully-connected layers and dense layers, because they are the one that tend to cause overfitting.

c)`
d)

1:https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939#:~:text=A%20CNN%20typically%20has%20three,and%20a%20fully%20connected%20layer.
2:https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243
3: Regularization technique CNN and Keras:
https://medium.com/intelligentmachines/convolutional-neural-network-and-regularization-techniques-with-tensorflow-and-keras-5a09e6e65dc