1. I am a graduate student about to complete my master's program in Applied Computational Intelligence, and I've been working as a Machine Learning Engineer for the past two years. I am experienced in solving natural language processing and computer vision tasks using machine learning.

2. I was looking for a meaningful project idea for my dissertation thesis when I stumbled upon this competition. The problem seemed challenging and I knew the work could contribute to a great cause, so I decided to compete and write my thesis around it.

3. **Data selection**
   Due to hardware limitations and also the high class imbalance in the full training data set, I only worked with a selected portion of the data. The strategy consisted in selecting only data from tier1 based on the project id, with the intuition that this might help model the diversity present in the large data set. The final experiments were conducted using the *micro* subset merged with subsets of at most 500 samples from each project, depending on availability. The final train set consisted of 7,931 videos, all from tier1, with 91.1% of them being flowing.

   Data preprocessing and augmentation
   In terms of preprocessing, the areas outlined in orange from each video are cropped and then resized to 112x112px. All three color channel of the videos are used, but before feeding the data into the neural networks, there are the usual feature scaling and standardization steps. Throughout the training phase, two types of data augmentations are performed on an entire video with a 25% chance each: rotations and flips.

   Models
   The final experiments were conducted with three different spatio-temporal neural network architectures based on ResNet-18: *R3D*, *MC3*, and *R(2+1)D*. They were designed for video classification and pretrained for action recognition on the Kinetics-400 data set. The decision layer of each network was replaced with a single-neuron layer with sigmoid activation. The models were trained using the Adam optimizer, with a batch size of 1 video, and the binary cross-entropy loss, with class weights of 1 for flowing samples and 10.23 for stalled ones, to compensate for the class imbalance. The initial learning rate is set to , and was decreased over time using cosine annealing. Due to hardware limitations, the models were only trained for 30 epochs.

   Results
   A decision threshold higher than 0.5 was selected empirically and used for each model. The best result obtained using a single model was the one of the *R(2+1)D* model. But, the highest performance was achieved using an ensemble of two *R(2+1)D* models and an *MC3* model. For the ensemble, a sample was considered to be stalled if at least two of the three constituent models had a decision level higher than their corresponding thresholds.

4. a. Positive (stalled) samples are given higher weight because they are not as well represented in the training set as negative (flowing) samples. Using class weights has made training the models possible despite the high class imbalance of 10:1 (negatives to positives).

```
# compute class weight for stalled samples
w = len(train_df[train_df["stalled"] == 0]) / len(train_df[train_df["stalled"] == 1])

# during training
if y[0].item() == 1:
    loss = loss * w
```

b. Augmentations had a great impact in narrowing the gap between the MCC score obtained on the validation set and the MCC score on the test set.

```
# define augmentations
self.rotations = [cv2.ROTATE_90_CLOCKWISE, v2.ROTATE_90_COUNTERCLOCKWISE,
cv2.ROTATE_180]
self.flips = [0, 1, -1]

# during training
if self.aug:
    if random.random() <= 0.25:
        rot = random.choice(self.rotations)
        x = [cv2.rotate(img, rot) for img in x]
    if random.random() <= 0.25:
        fl = random.choice(self.flips)
        x = [cv2.flip(img, fl) for img in x]
```

c. Using decision thresholds higher than 0.5 at evaluation has resulted in much better performance.

```
# define threshold
T = 0.95

# apply threshold
s = [0 if p < T else 1 for p in df["p"].values]
df["stalled"] = s
```

5. Machine specs and time:
    - CPU (model): Intel(R) Xeon(R) CPU @ 2.00GHz
    - GPU (model): NVIDIA Tesla P100 (16GB card)
    - Memory (GB): 13GB
    - OS: Linux
    - Train duration:
        - R(2+1)D model: ~51h
        - MC3 model: ~36h
        - R3D model: ~30h
        - Total: ~117h
    - Inference duration (for the entire test set):
        - R(2+1)D model: ~40m
        - MC3 model: ~30m

- R3D model: ~35m
- Total: ~105m

6. Things that didn't have a positive impact on model performance:
   - using positive (stalled) tier2 data with high confidence
   - removing the background around the region of interest after cropping it
   - using the crowd scores as targets instead of the binary labels

7. No.

8. I've also analyzed the accuracy, precision, recall, f1-score, and confusion matrix of the models on the validation set.

9. The models are quite large and do require the specified amount of GPU memory for training.

10. I do have some plots from the training process of the best single model - the R(2+1)D model:

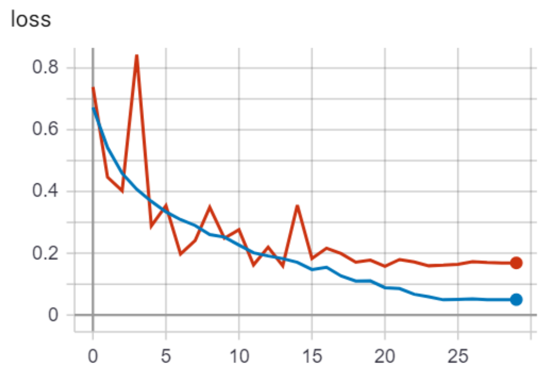

Figure 1. R(2+1)D model's loss. Blue: BCE loss on the training set.
Red: BCE loss on the validation set.

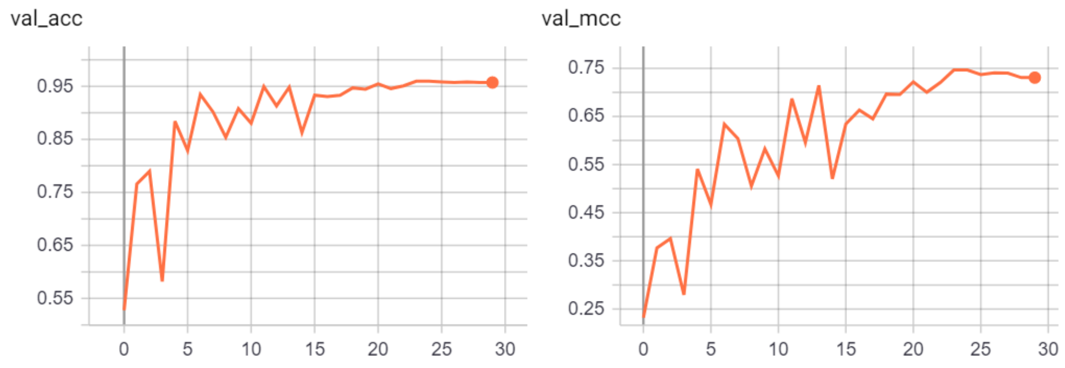

Figure 2. R(2+1)D model's accuracy and Matthews correlation coefficient.
Left: Accuracy on the validation set. Right: MCC score on the validation set.

11. As further work, I would like to experiment with larger samples of the training data, which would cause an even greater data imbalance than the 10:1 of my final experiments. This would most likely improve the performance of the models, and allow me to test the limits of training models with highly imbalanced data. Performing these experiments would require either or both better hardware and months of work.