**Solution Requirements**

1. The solution must be fully integrated into the ICGT. This means it must be an injective replacement for either one or more routines in the ICGT, such that other routines cannot tell the difference. Therefore, the solution must use the exact same data structures as in the rest of the code.
2. The solution will predict the evolution vector for each fracture tip at each time step, by first predicting the SIF. The predicted SIFs will be decomposed and compared against the ICGT calculations, whence it must achieve [give percentage][] agreement. The predicted vectors will then be analysed as magnitudes and angles, where each must be in [give percentage][] agreement with the ICGT calculations.
3. This prediction will be achieved by using machine learning techniques trained upon prior runs of ICGT.
4. Initially, the evolution of each fracture tip will be predicated only by its own history, encoded in past values for location, displacement, and SIF.
5. The first expansion objective is to incorporate more history into each predictive cycle, from one step back to two steps or three steps back etc… It is hoped that this would increase accuracy.
6. The second expansion objective is to incorporate a radius of influence for each fracture tip, where the information from other fracture tips falling within this radius is used in the prediction for the tip in question.

**Solution Design**

[create diagram][]

**High Level:**

On the broadest level, the problem posed is to take a fracture tip and its position, previous displacement and previous SIF (with more features possibly being added in expansion), and then predict its future SIF and therefore evolution.

It is then clear that there are two functions that must be performed by the solution; first, to predict the SIF, and second, to use the SIF to predict evolution. Of these, the former is significantly more complex and time consuming during the operation of the ICGT, whereas the latter is rather well understood and efficient. As the purpose of the former is to predict information already generated by the ICGT, and the purpose of the latter is to use this information, the two functions are actually wholly independent with respect to design and operation (though of course the prediction returned by the latter is itself wholly dependent on the result of the former if they are operated in series).

Therefore, this solution will hereafter be split into two components, one which is responsible for the prediction of the SIF (Predictor – Stress Intensity Factor, P-SIF), and one which is responsible for the prediction of the fracture tip evolution vector (Predictor – Tip Evolution Vector, P-TEV) given the results of P-SIF.

The calculation of the SIF in the ICGT is a very involved process, requiring the solution of several PDEs. This strategy will not require this. Instead, it will use a machine learning algorithm trained on features of the fracture in order to predict the SIF.

The features used initially for each fracture tip will be the current location of the tip, its previous displacement, and its previous SIF (in the future it may also use more of its own history, as well as the information from other fracture tips within a defined radius of influence). Given these features it will then predict the three components of the SIF independently, namely and.

Following the calculation of the SIF of a given tip (it is irrelevant whether this is by the normal ICGT manner or the new P-SIF strategy), P-TEV will then predict the evolution vector of the tip. This is a vector in three dimensions, and the element of the vector in each dimension will be predicted independently. The features used will be the same as for P-SIF, except that the newly predicted SIF will take the place of the previous SIF.

**Low Level:**

Before calling either the P-SIF or P-TEV routine (jointly known as predictor routines) themselves, a wrapper will be called first. The purpose of this wrapper is twofold – to package information from the wider program into a format that the routines can use efficiently, and then to parse the return from the routines such that the wider program can continue with the information generated. The wrapper for each routine will be largely similar, although it is expected that there will be minor differences.

Called inside the wrapper will be the predictor routine itself. It is currently expected that the machine learning strategy to be implemented will be an Artificial Neural Network (ANN), though this is liable to change during the testing that will occur prior to full implementation.

It is readily apparent that training the network whilst integrated into the ICGT would be wildly inefficient, due to all the superfluous calculations being performed. Therefore, the training stage of the network, where the weights and biases will be assigned, will be performed in isolation. In order to do this, a training set must be created. This will be done by extracting data from several runs of the ICGT, namely the values for the features of a tip, its final SIF for that time step, and its final location for that time step. This will therefore be a labelled set. This set will then be divided into training and validation sets for the training process. A test set will not be constructed – instead, testing will occur whilst the trained network is integrated into the ICGT, so that the test set will always be novel data [is this wise? Maybe do this with validation instead as an anti-overfitting measure? Discuss][].

As both the SIF and TEV have three components, each component will have to be predicted separately. This could be done with either one network with three outputs, or three networks with one output each. These two options will be subject to testing to determine which is better (it is expected that there will be an accuracy/efficiency trade-off).

Once the network has been trained, the network will be imported into the predictor routine. At this point, the P-SIF routine would be expected to be able to predict the SIF, and the P-TEV to predict the TEV, although the accuracy to which they can do this is unknown, presumed poor. This is the point at which testing will begin.

The testing regime will be largely based on the identification of which training hyperparameters to use. This is a non-trivial task and has its own section.

**P-SIF Low Level:**

[direct knowledge of incoming/outgoing data structures for wrapper, knowledge of input/output to/from P-SIF][]

[variable names etc][]

[actual neural network designs – neurons, layers, outputs etc][]

[preliminary alternate ML strategies][]

**Testing Strategy:**

There are four main objectives of the testing strategy: first, to determine whether the predictor routines are actually successfully predicting the desired quantities; second, to identify which machine learning algorithm is most suited to this task (currently expected to be an ANN); third, to decide whether one network with three outputs or three networks with one output each is preferable; and finally fourth, once all these other questions are answered, to determine the optimal set of hyperparameters for the algorithm.

To answer the first testing requirement, the results generated by the predictor routines will be compared to the results calculated by the ICGT. In the case of the SIF, each component will be compared in isolation. In the case of the TEV, two quantities will be compared – the magnitude of the vector and the angle of the vector. The test will be considered a success if the lowest agreement percentage from all these measures across several tests is greater than 70%.

The second testing objective is significantly more challenging, as it would require vast quantities of testing across multitudes of hyperparameters to ensure the correct answer. As this level of testing is unfeasible, a more heuristic approach will be taken. As the expected most efficient strategy is an ANN, this will be the baseline, where it will be run with a limited number of hyperparameter configurations. Then, for each of these configurations, another machine learning algorithm (e.g. a CNN) will be run with as close to an identical set of hyperparameters as possible. At this stage, the speed and accuracy of the results will be inspected. If the performance of an alternate strategy is promising, there will be further testing. This is a heuristic approach because whether a given strategy is promising or not will come down to judgement.

The third objective is relatively simple by comparison, as it has a very well-defined problem. For each of the two predictors, a number of tests with differing hyperparameter configurations will be run for each possibility (i.e. one network three outputs versus three networks one output each). Then, given the results of this testing, the optimal strategy will be selected. It should be noted that it is expected that neither method will be exclusively superior to the other, and therefore a judgement call will once again be needed.

The final testing objective is arguably the most important, as the exact hyperparameter configuration of the model will have massive ramifications for the accuracy and efficiency of the final solution. Incorporated into this stage will be any degree of data augmentation that will be added to the training set. Strategies for data augmentation are discussed elsewhere [write this][]. Beyond data augmentation, there are still several critical hyperparameters that need to be tuned, including the specific architecture of the network itself, as well as all the other quantities that a neural network requires.

As this is by far the most computationally intensive testing objective with the most degrees of freedom, the testing regimen for this objective must be more rigorously defined than that which was appropriate for the others. The regimen here will have to be largely heuristic for the network architecture and augmentation strategy, as it was for the other testing objectives. However, for the remaining hyperparameters, a more complex strategy can be invoked. Whilst the behaviour of the solution accuracy with respect to the hyperparameters is highly non-linear, it should be able to be approximated as continuous, and therefore gradient descent should be a valid strategy. Of course, this will be gradient descent in many dimensions, and therefore certain hyperparameters may have to be excluded, or the total quantity of degrees of freedom otherwise reduced in some manner. If this does prove to be too computationally intensive to be feasible, gradient descent would be replaced by random selection from each hyperparameter’s domain (random selection is more efficient than a grid search [cite this maybe?][]).

With the conclusion of these testing stages, the final prototype should be ready for analysis.