Predicting congested areas with recurrent neural networks

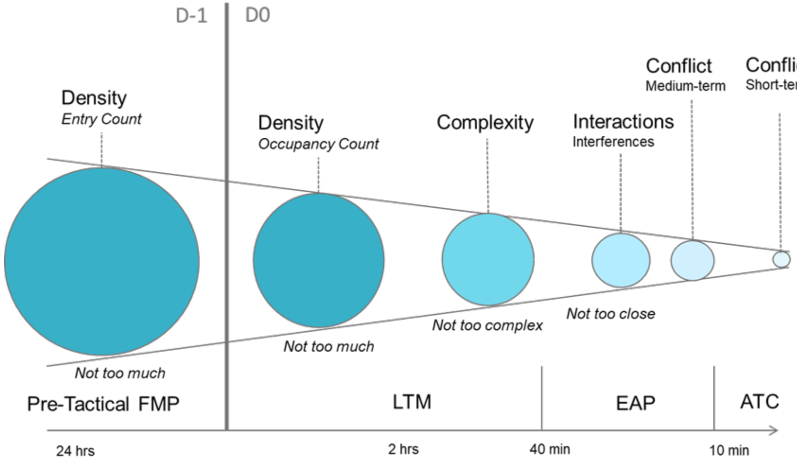
Loïc Shi-Garrier

December 15, 2020

Introduction

- ATFCM is looking at occupancy problems: balance demand/capacity
- ATC is looking at proximity problems: ensure separation of aircraft
- There is a gap between strategic and tactical operations

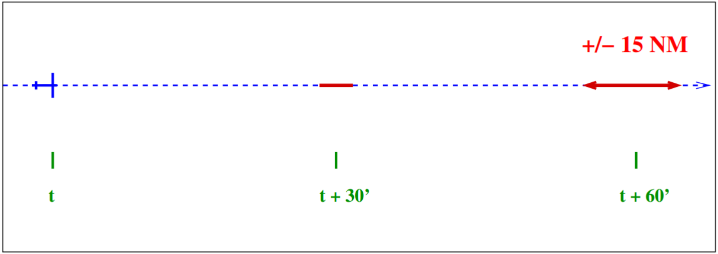
▶ Formation of "hot spots"



Introduction

Subject: *Predicting congested areas with recurrent neural networks*

- Timeframe: 40 minutes before crossing the sector
- Uncertainties are too high to predict which trajectories will be involved in conflicts -> predict congested areas
- Recurrent neural networks: well suited to process sequences like trajectories



Content

Content

Complexity metric

Goal: to quantify the "complexity" of a traffic situation

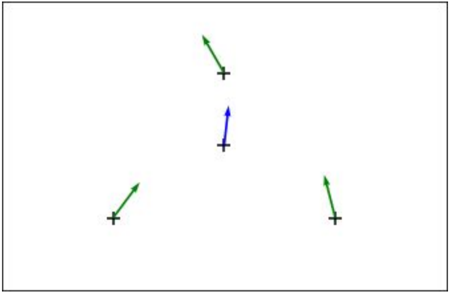
The complexity is linked with:

- The workload of the ATCO
- The conflicts probability
- The geometry of the traffic (convergence/divergence) and of the airspace

- We use a complexity metric based on linear dynamical systems

Complexity metric

Let $X = \begin{bmatrix} x_1 & x_2 & \cdots \\ y_1 & y_2 & \cdots \\ z_1 & z_2 & \cdots \end{bmatrix}$ and $\dot{X} = \begin{bmatrix} v_{x_1} & v_{x_2} & \cdots \\ v_{y_1} & v_{y_2} & \cdots \\ v_{z_1} & v_{z_2} & \cdots \end{bmatrix}$

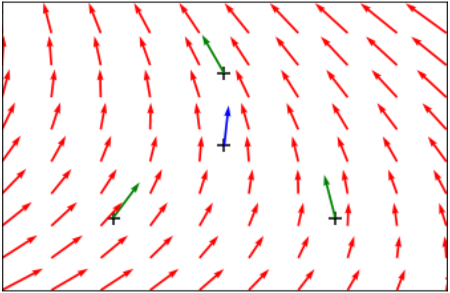


Complexity metric

Let $X = \begin{bmatrix} x_1 & x_2 & \cdots \\ y_1 & y_2 & \cdots \\ z_1 & z_2 & \cdots \end{bmatrix}$ and $\dot{X} = \begin{bmatrix} v_{x_1} & v_{x_2} & \cdots \\ v_{y_1} & v_{y_2} & \cdots \\ v_{z_1} & v_{z_2} & \cdots \end{bmatrix}$

We consider the linear dynamical system $\dot{X} \approx AX + b$

A and b are obtained with least squares



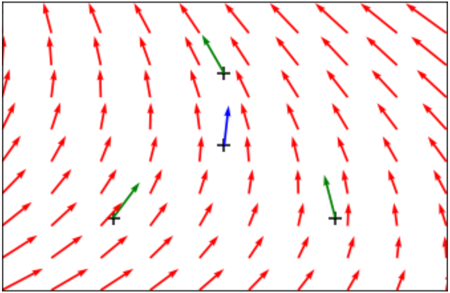
Complexity metric

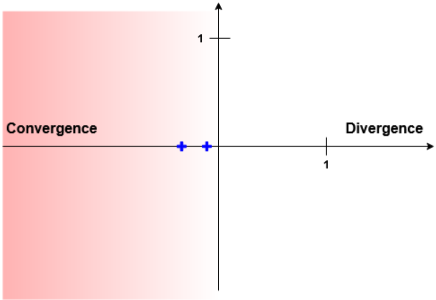
Let $X = \begin{bmatrix} x_1 & x_2 & \cdots \\ y_1 & y_2 & \cdots \\ z_1 & z_2 & \cdots \end{bmatrix}$ and $\dot{X} = \begin{bmatrix} v_{x_1} & v_{x_2} & \cdots \\ v_{y_1} & v_{y_2} & \cdots \\ v_{z_1} & v_{z_2} & \cdots \end{bmatrix}$

We consider the linear dynamical system $\dot{X} \approx AX + b$

A and b are obtained with least squares

The metric is defined as $C = \sum_{\mathrm{Re}(\lambda(A))<0} |\mathrm{Re}(\lambda(A))|$





Recurrent Neural Networks

- RNN are used to process sequences of variable lengths
- Basic RNN cell: $y_t = \phi(W \cdot [x_t; y_{t-1}] + b)$
- Many applications in Natural Language Processing
- Many variants like LSTM and GRU

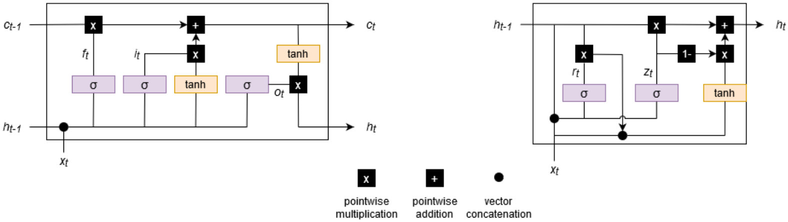


Figure 1: LSTM layer (left) and GRU layer (right)

Content

Dataset

A trajectory is a sequence of aircraft states $\begin{bmatrix}\theta & \phi & h & GS & HDG & Vz\end{bmatrix}^T$.

- 8,011 simulated trajectories
- 3,025 time steps (1 time step = 15s)
- 2,434 sequences of length 160 for training
- 271 sequences of length 160 for testing

We define a supervised learning regression task.

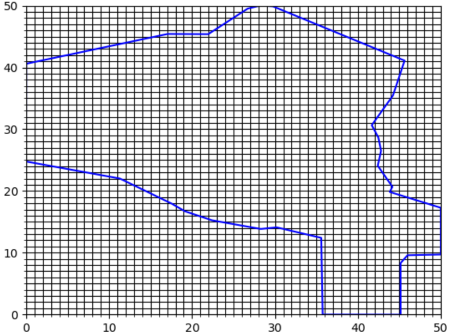
Definition of the inputs

For each time step t, there is a set of aircraft states
$state_1 = \begin{bmatrix}\theta_1 & \phi_1 & h_1 & GS_1 & HDG_1 & Vz_1\end{bmatrix}^T \;;\; state_2 \;;\; \cdots$
These states are concatenated in $x[t] = \begin{bmatrix}state_1 & state_2 & \cdots\end{bmatrix}^T$
The input is defined as the sequence $X_{t_0} = \left(x[t_0], \; \cdots, \; x[t_0 + T - 1]\right)$

Dataset

Definition of the target outputs

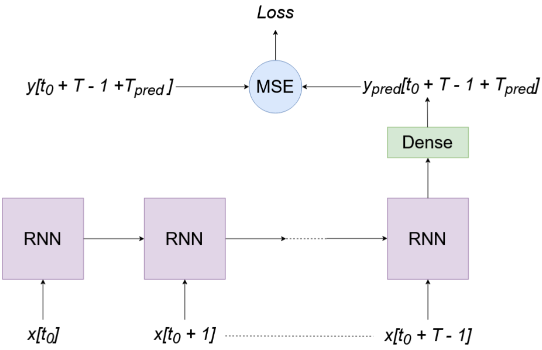
For each time step t, we define a $N \times N$ matrix $y[t]$
For each cell (i, j), $y[t]_{i,j}$ is the maximum value of the complexity metric inside this cell (rescaled by a logarithmic function)



- The input X_{t_0} is associated with the output $y[t_0 + T - 1 + T_{pred}]$

A first model

The model uses the last T traffic situations to predict the values of the complexity metric in every area of the airspace in T_{pred} time steps.



Hyperparameters

Default hyperparameters:

- $N = 100$
- $T = 160$
- $T_{pred} = 160$
- class of recurrent layer $rl = GRU$
- $l = \begin{bmatrix} 512 & 512 & 512 & 512 \end{bmatrix}$
- batch size $bs = 64$
- learning rate $lr = 10^{-4}$
- dropout rate $dr = 10^{-1}$
- L2 regularization coefficient $lbd = 10^{-5}$
- gradient norm scaling value $cn = 1$
- nb of epochs $= 300$

Sensitivity analysis

Hyperparameters values	Validation loss ($\times 10^{-4}$)
Default	180
$T = 80$	182
$T = 320$	188
$rl = LSTM$	154
$l = [256, 256, 256, 256]$	168
$l = [1024, 1024, 1024]$	191
$l = [512, 512, 512, 512, 512, 512]$	213
$l = [1024, 256, 256, 1024]$	189
$bs = 32$	158
$bs = 128$	232
$lr = 10^{-5}$	720
$lr = 10^{-3}$	112
$dr = 0$	176
$dr = 0.3$	202
$lbd = 0$	102
$lbd = 10^{-4}$	312
$cn = 10^{-1}$	189
$cn = 10$	198

Table 1: Validation loss averaged over the 10 last epochs for various hyperparameters values

Results

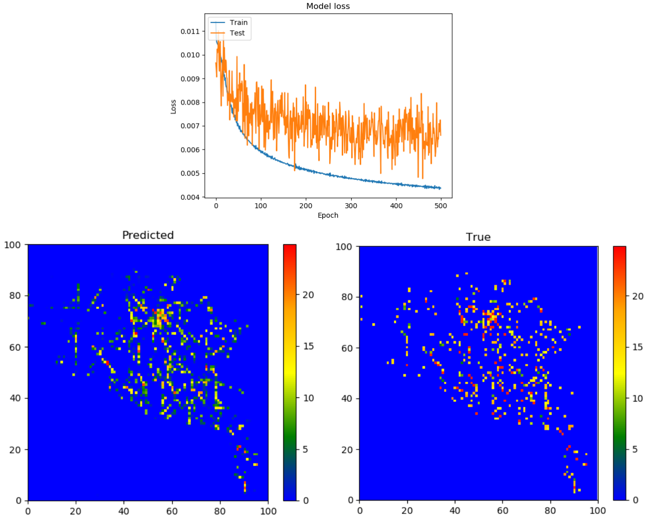
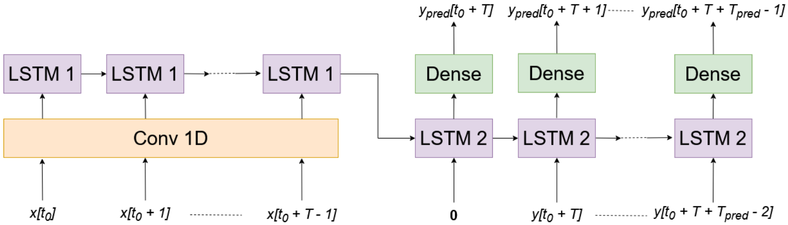


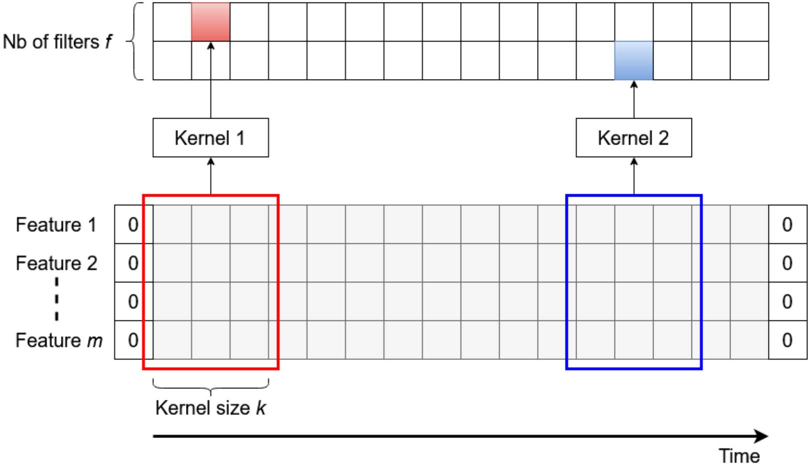
Figure 2: Model losses (top) and validation example with predicted values (left) and true values (right)

A second model: encoder-decoder network

- Encoder-decoder model with "teacher forcing"
- Contrary to the previous model, the entire sequence is predicted (not only the last element)
- During inference, the input $y[t]$ is replaced by the prediction $y_{pred}[t]$ of the previous step



Convolutional layer



Preprocessing with Gaussian smoothing

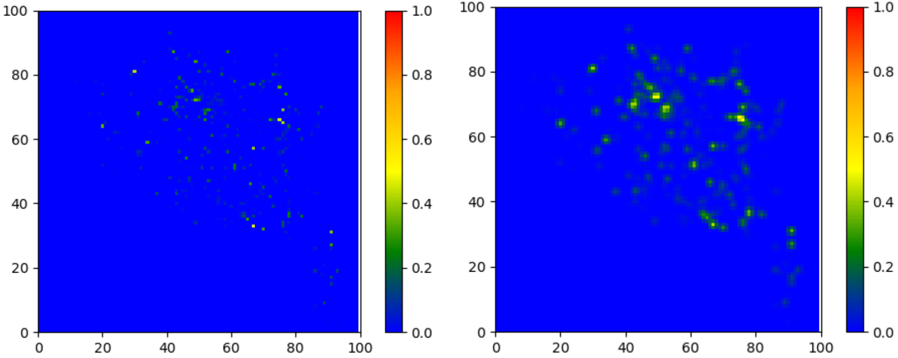


Figure 3: Original $y[t]$ (left) and smoothed $y[t]$ (right)

Hyperparameters

- $N = 100$
- $T = 160$
- $T_{pred} = 160$
- nb of LSTM hidden units $h = 128$
- nb of Conv filters $f = 512$
- Conv kernel size $k = 3$
- batch size $bs = 128$
- learning rate $lr = 10^{-3}$
- nb of epochs $= 100$

Results

Validation set: 271 examples (2,710,000 values)

182,982 non-zero values with mean = 0.4

678,240 non-zero values with mean = 0.08

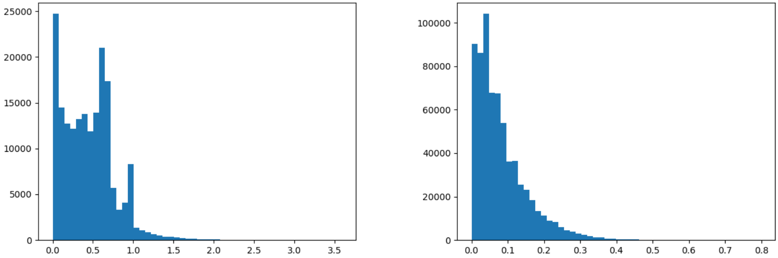


Figure 4: Distribution of non-zero absolute errors over the validation set with default model (left), and with Gaussian smoothing (right)

Example

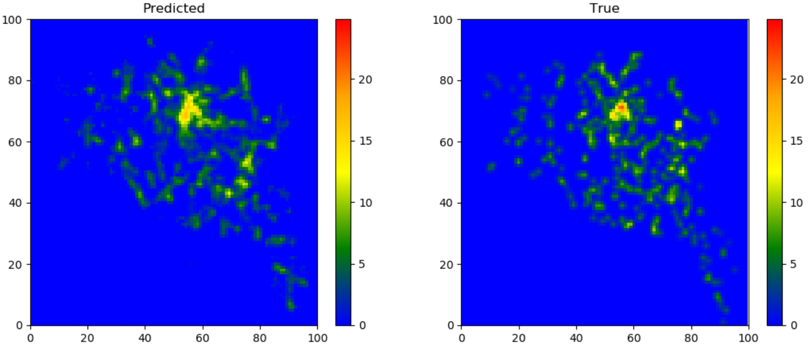
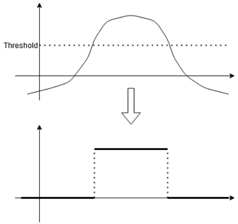


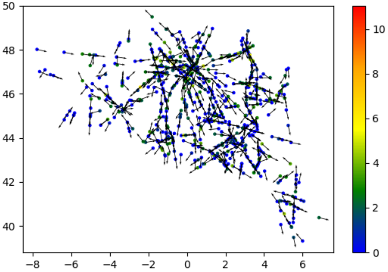
Figure 5: Validation example with predicted values (left) and true values (right)

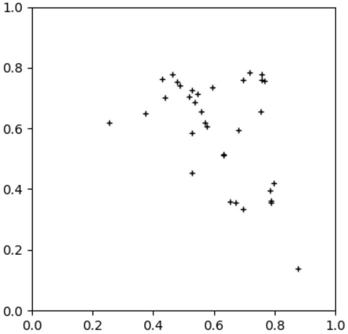
Content

Definition of congested areas

Given a time step t, we have a set of aircraft, with a complexity value for each aircraft.

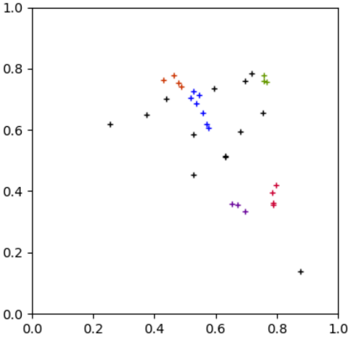




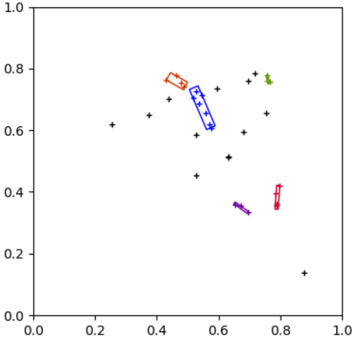


Definition of congested areas

A clustering algorithm (DBSCAN) is applied to detect clusters of aircraft with high complexity.



For each cluster, a box is defined with 5 parameters: center point coordinates, width, height and rotation angle.



Definition of congested areas

A $N \times N$ grid is defined over the airspace.

Each cell of the grid is defined as a vector of dimension 6:

$$\begin{bmatrix} p & x & y & w & h & \Phi \end{bmatrix}$$

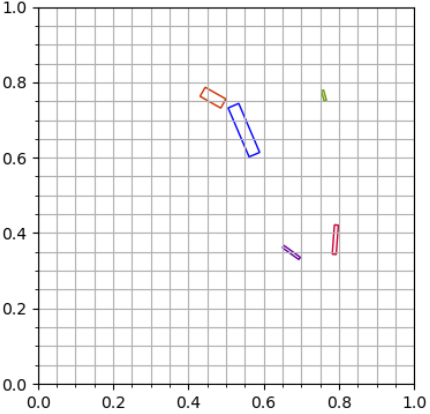


Figure 6: The target output $y[t]$ is a $N \times N \times 6$ tensor

Model

- Predict only the congested areas at the 40th minute (not the entire sequence)
- 2 LSTM layers followed by 2 dense layers
- Loss function: $L(y, y_{pred}) = \sum_i 1_i(x_i - x_{i,pred})^2 + 1_i(y_i - y_{i,pred})^2 + 1_i(\sqrt{w_i} - \sqrt{w_{i,pred}})^2 + 1_i(\sqrt{h_i} - \sqrt{h_{i,pred}})^2 + \xi 1_i(p_i - p_{i,pred})^2 + (1 - 1_i)(p_i - p_{i,pred})^2$
- $\xi = 5$

Results

Training set: Precision = 0.63 ; Recall = 0.60 ; MCC = 0.61
Validation set: Precision = 0.21 ; Recall = 0.31 ; MCC = 0.25 ;

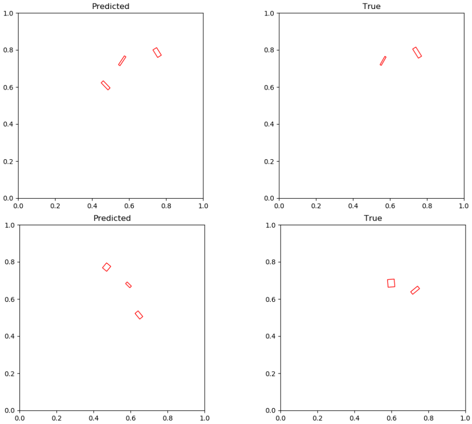


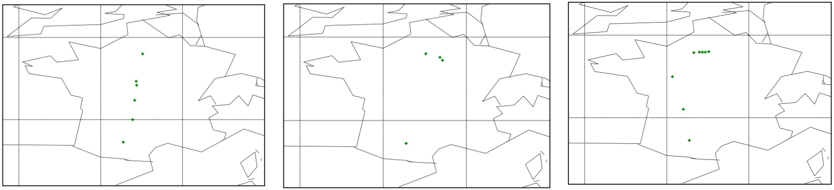
Figure 7: Prediction on the training set (top) and on the validation set (bottom)

Content

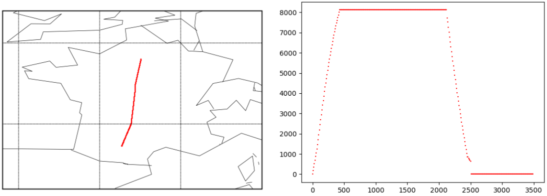
Dataset

- 3 Flight Plans from TLS to CDG
- 8 grib2 files over two days containing wind speed information
- 862 generated trajectories

The Flight Plans are defined as sequences of 2D points.



A point is generated every 10 seconds



Model: encoder-decoder network

- The flight plan is provided to the encoder as a sequence of 2D points
- The states inputted in the decoder contain the 3D position and the two wind components (closest to the current position)
- The decoder predicts the 3D position
- The encoder and the decoder are both composed of 2 layers of LSTM with hidden dimension 256
- There are 3 dense layers with output dimensions: 128, 64, and 3

Results

Proportion of true trajectory as input	MSE on validation set
100 %	1.1×10^{-3}
75 %	1.9×10^{-3}
25 %	2.267×10^{-1}

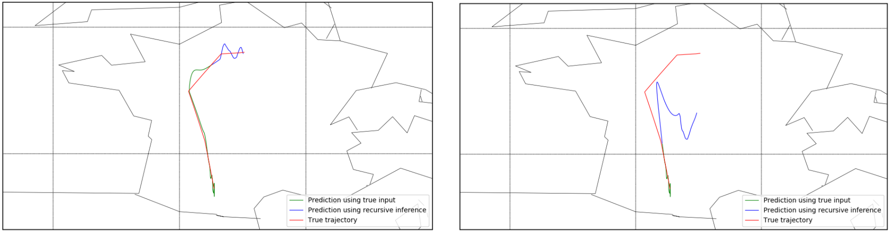


Figure 8: Proportion of true trajectory as input: 75% (left) and 25% (right).

Conclusion

- Predict the complexity: the encoder-decoder model provides good results when using Gaussian smoothing on the output data
- Detect congested areas: the model requires fine-tuning of its parameters to achieve reasonable performances on the validation set
- Trajectory prediction: the encoder-decoder model is unable to generate meaningful predictions in inference mode

Future work

- Predict the estimated time of overflight to selected waypoints and use these predictions to compute a measure of complexity -> more relevant from an operational perspective
- Estimate the uncertainty of the predictions by propagating the covariance (with Bayesian networks)
- Suggest actions to mitigate the hot spots

Questions