Australian Government
Bureau of Meteorology

Accelerated Computing with Convolutional Autoencoder Networks

Justin Freeman
Bureau of Meteorology

Concept

Use a convolution neural network to predict the pressure solution in a fluid dynamics model.

Reduce the iterative workload of the numerical solver

Training data

Initial Pressure and converged Pressure solution

Method

Convolutional autoencoder network

Implementation

Neural Network predicts converged solution,
CFD software proceeds as usual

CFD model

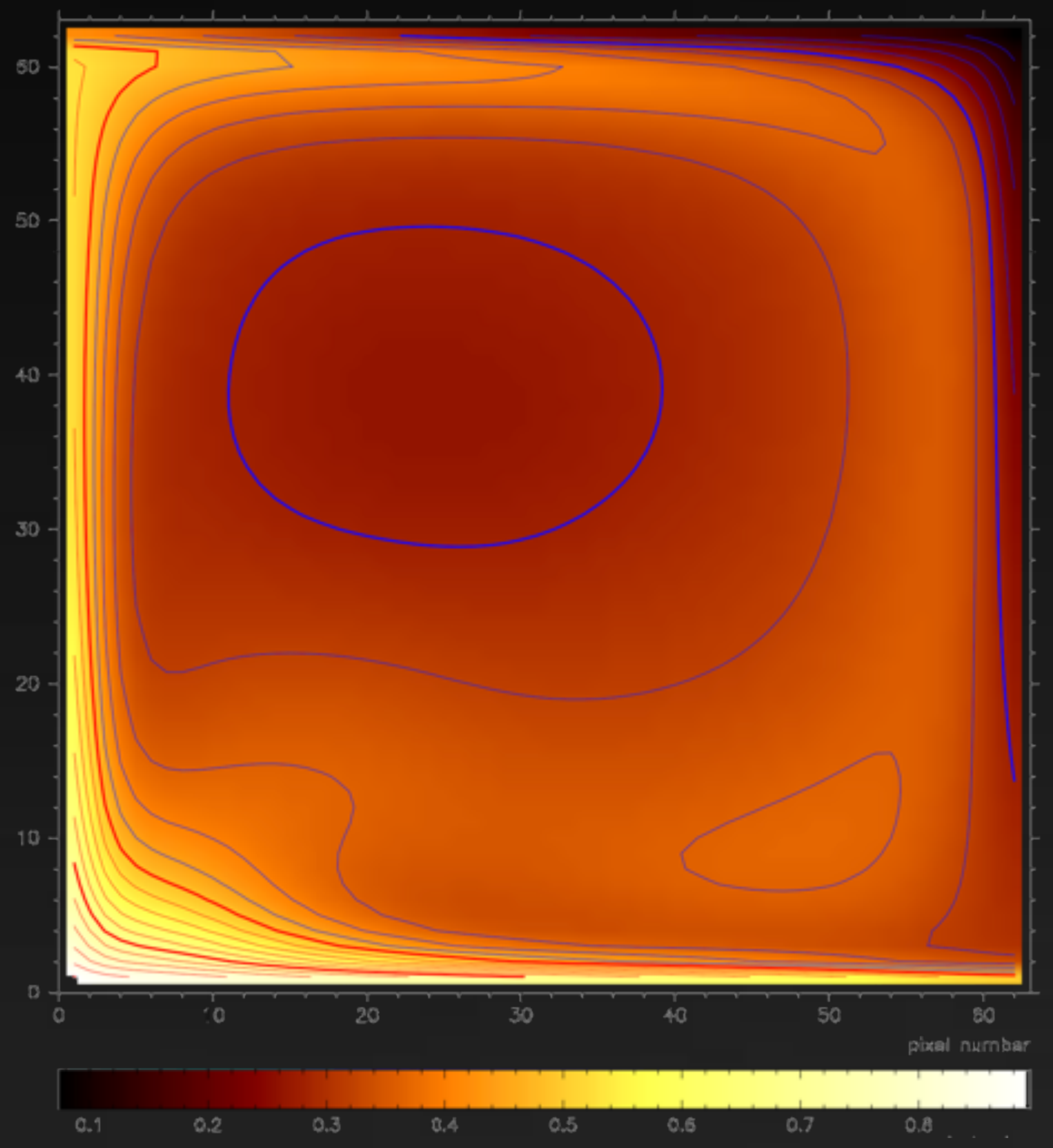
Rayleigh-Bernard Convection

instabilities caused by rising hot fluid and falling cold fluid

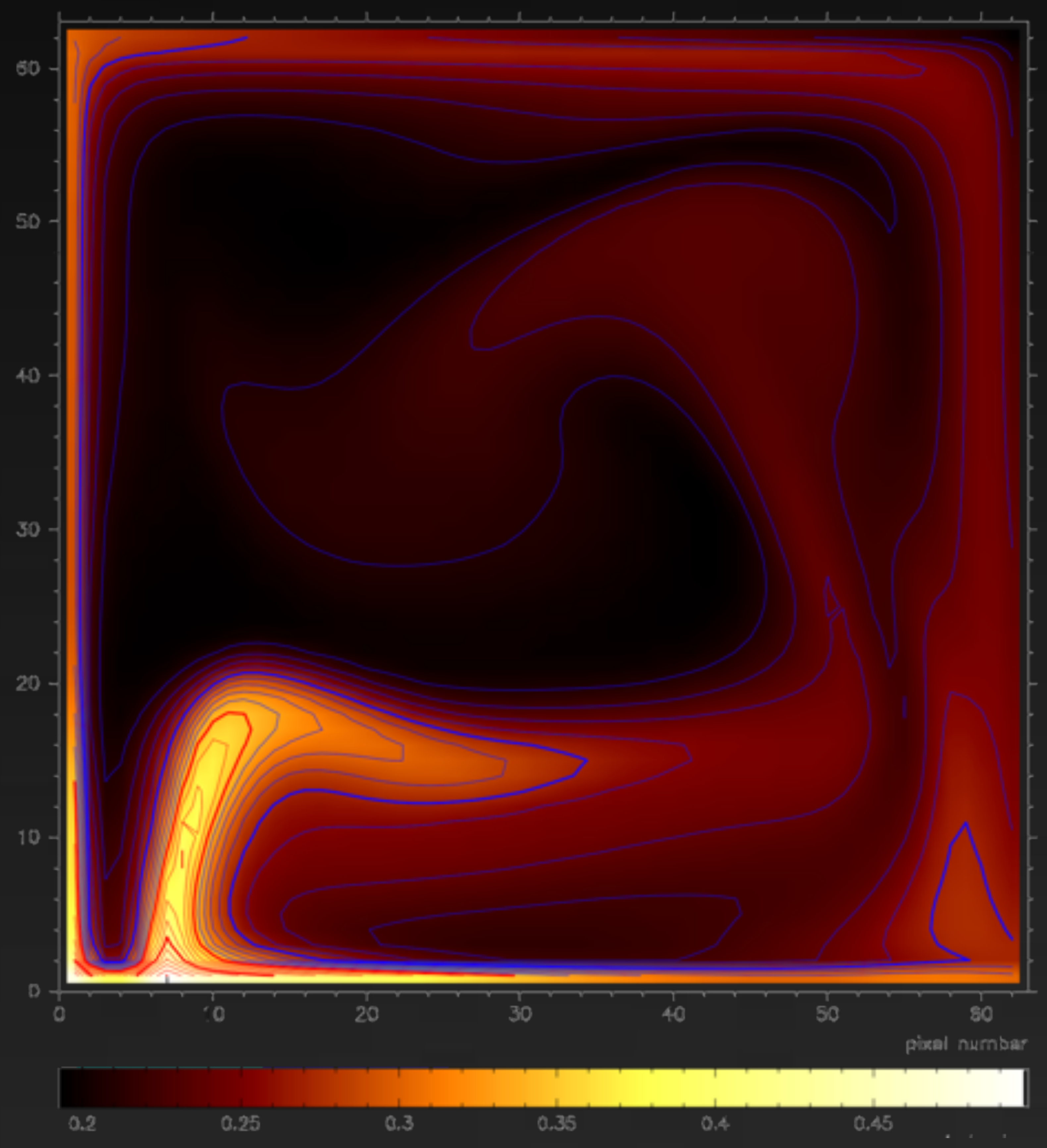
$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u + \nabla p = \frac{1}{Re}\nabla^2 u + f$$

$$\nabla \cdot u = 0$$

$$\frac{\partial T}{\partial t} + u \cdot \nabla T = \alpha \nabla^2 T + q$$



Ra = 1x10^6



Ra = 1x10^9

CFD model

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u + \nabla p = \frac{1}{Re}\nabla^2 u + f$$

$$\nabla \cdot u = 0$$

$$\frac{\partial T}{\partial t} + u \cdot \nabla T = \alpha \nabla^2 T + q$$

The algorithm is

1. Compute F^n, G^n from the velocities u^n and v^n,

$$F = u^n + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + f_x \right]$$

$$G = v^n + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial y} - \frac{\partial (uv)}{\partial x} + f_y \right]$$

2. Solve the Poisson equation for the pressure p^{n+1},

$$\frac{\partial^2 p^{n+1}}{\partial x^2} + \frac{\partial^2 p^{n+1}}{\partial y^2} = \frac{1}{\delta t}\left(\frac{\partial F^n}{\partial x} + \frac{\partial G^n}{\partial y} \right)$$

3. Compute the new velocity field for u^{n+1} and v^{n+1} with the pressure field p^{n+1}.

$$u^{n+1} = u^n + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + f_x - \frac{\partial p}{\partial x} \right]$$

$$v^{n+1} = v^n + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial y} - \frac{\partial (uv)}{\partial x} + f_y - \frac{\partial p}{\partial y} \right]$$

Algorithm

Set t=0, n=0
Initialise variables u,v,p,T
While t<tend
    choose delta t (timestep)
    set boundary conditions for u, v, T
    compute T^{n+1}
    compute F^n and G^n
    compute RHS of pressure equation
    set it = 0
    while it < it_{max} and (residual norm > tolerance)
        perform SOR cycle
        compute residual norm of pressure equation
        it += 1
    compute u^{n+1} and v^{n+1}
    t += delta t
    n +=1

$$F = u^n + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + f_x \right]$$

$$G = v^n + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial y} - \frac{\partial (uv)}{\partial x} + f_y \right]$$

$$\frac{\partial^2 p^{n+1}}{\partial x^2} + \frac{\partial^2 p^{n+1}}{\partial y^2} = \frac{1}{\delta t}\left(\frac{\partial F^n}{\partial x} + \frac{\partial G^n}{\partial y} \right)$$

$$u^{n+1} = u^n + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + f_x - \frac{\partial p}{\partial x} \right]$$

$$v^{n+1} = v^n + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial y} - \frac{\partial (uv)}{\partial x} + f_y - \frac{\partial p}{\partial y} \right]$$

Algorithm

Set t=0, n=0
Initialise variables u,v,p,T
While t<tend

- choose delta t (timestep)
- set boundary conditions for u, v, T
- compute T^{n+1}
- compute F^n and G^n
- compute RHS of pressure equation
- set it = 0
- while it < it_{max} and (residual norm > tolerance)
  - perform SOR cycle
  - compute residual norm of pressure equation
  - it += 1
- compute u^{n+1} and v^{n+1}
- t += delta t
- n +=1

Inputs: A, b, ω
Output: ϕ

Choose an initial guess ϕ to the solution
repeat until convergence
  for i **from** 1 **until** n **do**
    $\sigma \leftarrow 0$
    for j **from** 1 **until** n **do**
      if $j \neq i$ **then**
        $\sigma \leftarrow \sigma + a_{ij}\phi_j$
      end if
    end (j-loop)
    $\phi_i \leftarrow (1-\omega)\phi_i + \frac{\omega}{a_{ii}}(b_i - \sigma)$
  end (i-loop)
  check if convergence is reached
end (repeat)

Profile

5200 steps at Ra = 5×10^6,
Pressure tolerance 1×10^{-9}

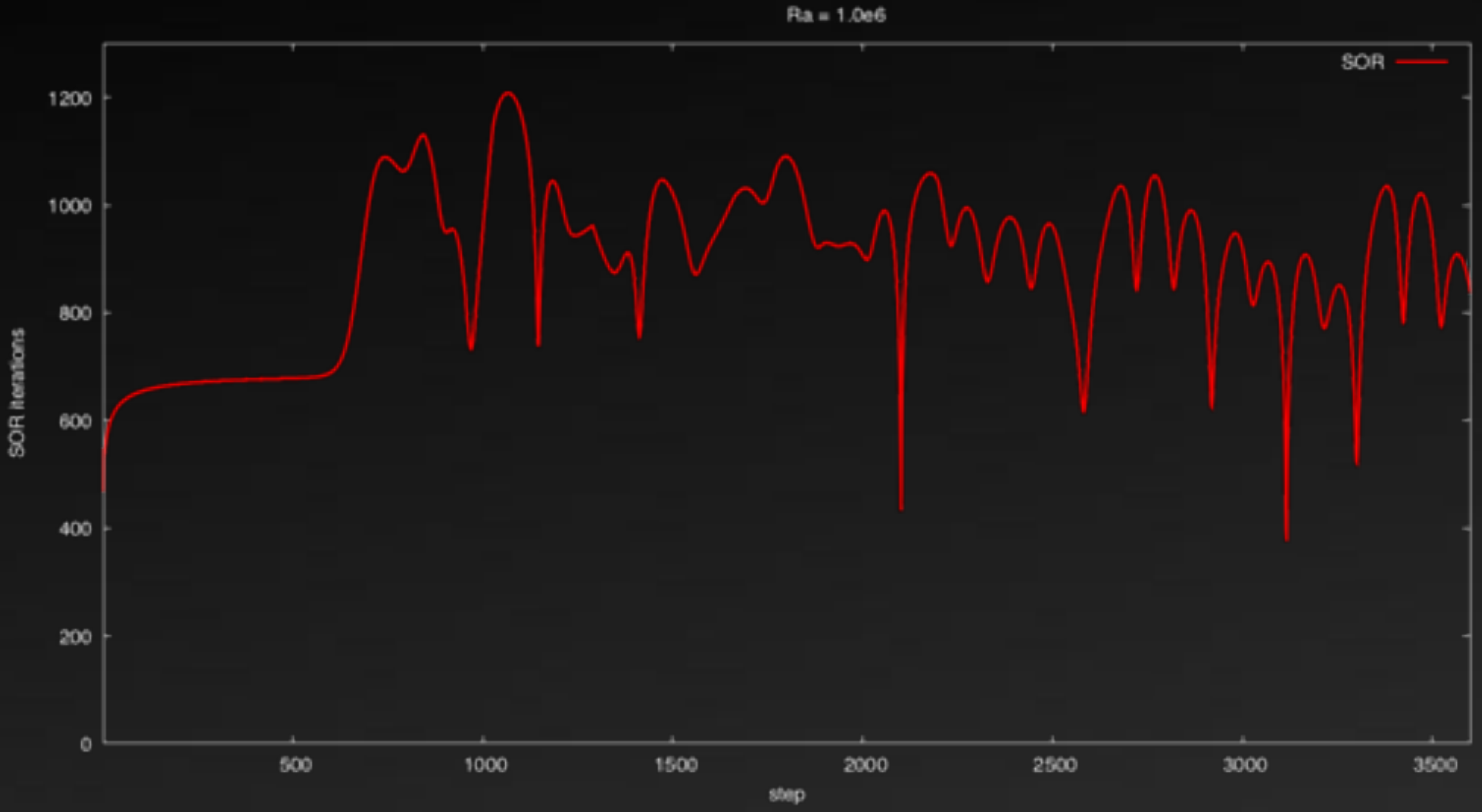
```
granularity: each sample hit covers 2 byte(s) for 0.01% of
96.54 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]     99.2   95.78    0.00                 poisson [1]
-----------------------------------------------
                                                 <spontaneous>
[2]      0.6    0.24    0.36                 comp_fg [2]
                0.09    0.00 19530248/19530248     d2u_dy2 [3]
                0.09    0.00 19530248/19530248     du2_dx [4]
                0.04    0.00 19530248/19530248     d2u_dx2 [7]
                0.04    0.00 19530248/19530248     duv_dy [9]
                0.04    0.00 19530248/19530248     duv_dx [8]
                0.03    0.00 19530248/19530248     d2v_dx2
[10]
                0.03    0.00 19530248/19530248     dv2_dy [11]
                0.00    0.00 19530248/19530248     d2v_dy2
[16]
-----------------------------------------------
                0.09    0.00 19530248/19530248     comp_fg [2]
[3]      0.1    0.09    0.00 19530248          d2u_dy2 [3]
-----------------------------------------------
                0.09    0.00 19530248/19530248     comp_fg [2]
[4]      0.1    0.09    0.00 19530248          du2_dx [4]
-----------------------------------------------
                                                 <spontaneous>
[5]      0.1    0.06    0.00                 comp_temp [5]
                0.00    0.00    5164/5164        RMATRIX [17]
-----------------------------------------------
```

Set t=0, n=0
Initialise variables u,v,p,T
While t<tend
- **choose delta t (timestep)**
- **set boundary conditions for u, v, T**
- **compute T^{n+1}**
- **compute F^n and G^n**
- **compute RHS of pressure equation**
- **set it = 0**
- **while it < it_{max} and (residual norm > tolerance)**
  - **perform SOR cycle**
  - **compute residual norm of pressure equation**
  - **it += 1**
- **compute u^{n+1} and v^{n+1}**
- **t += delta t**
- **n +=1**

99 %

Pressure Iterations

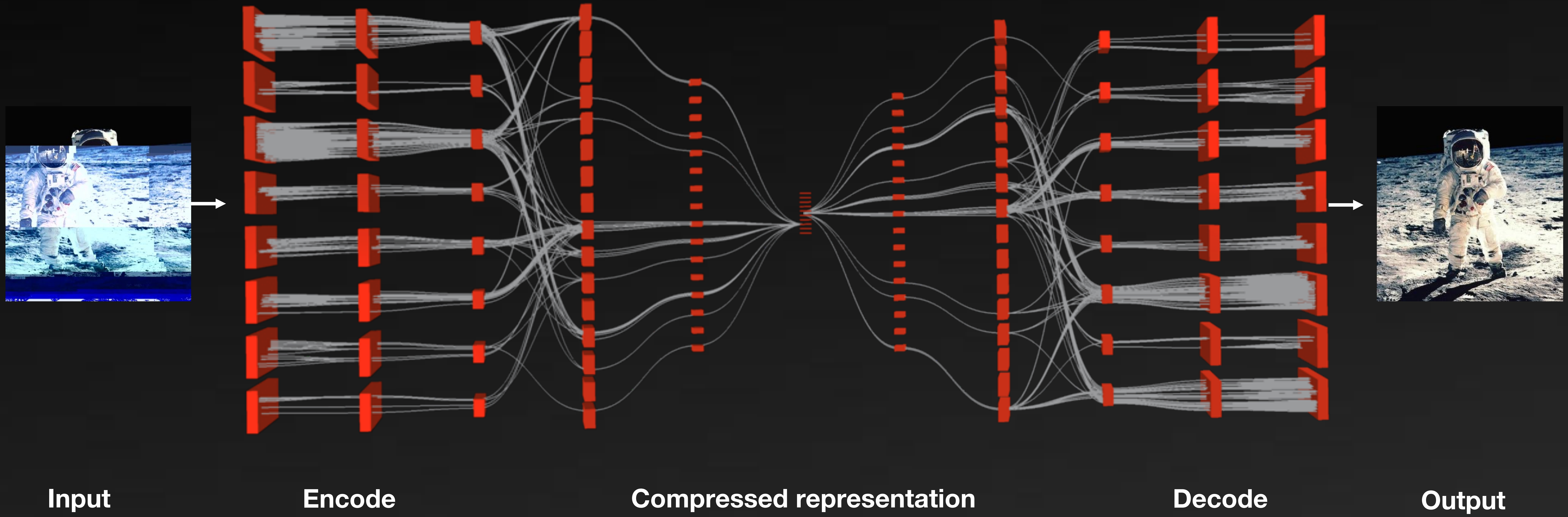


Pressure Iterations

shift the red line towards the green line using a neural network

ML model

Autoencoder network



Data

```
Set t=0, n=0
Initialise variables u,v,p,T
While t<tend
    choose delta t (timestep)
    set boundary conditions for u, v, T
    compute T^{n+1}
    compute F^n and G^n
    compute RHS of pressure equation
    set it = 0
    while it < it_max and (residual norm > tolerance)
        perform SOR cycle
        compute residual norm of pressure equation
        it += 1
    compute u^{n+1} and v^{n+1}
    t += delta t
    n +=1
```

Training Data

Save initial Pressure field

Save converged Pressure field

64x64 mesh

Run over a range of Ra:

3×10^6

4.9×10^6

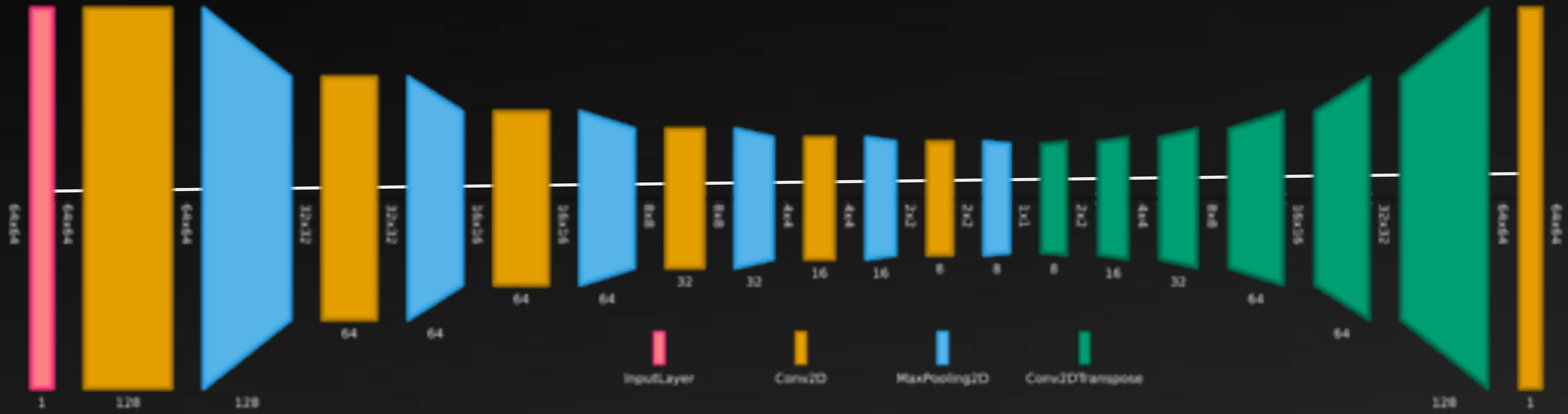
5×10^6

5.1×10^6

8×10^6

ML model

Autoencoder network



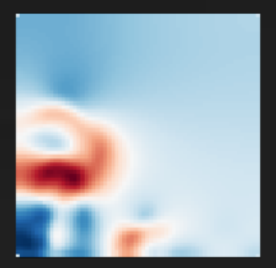
ML model Data Pre-processing

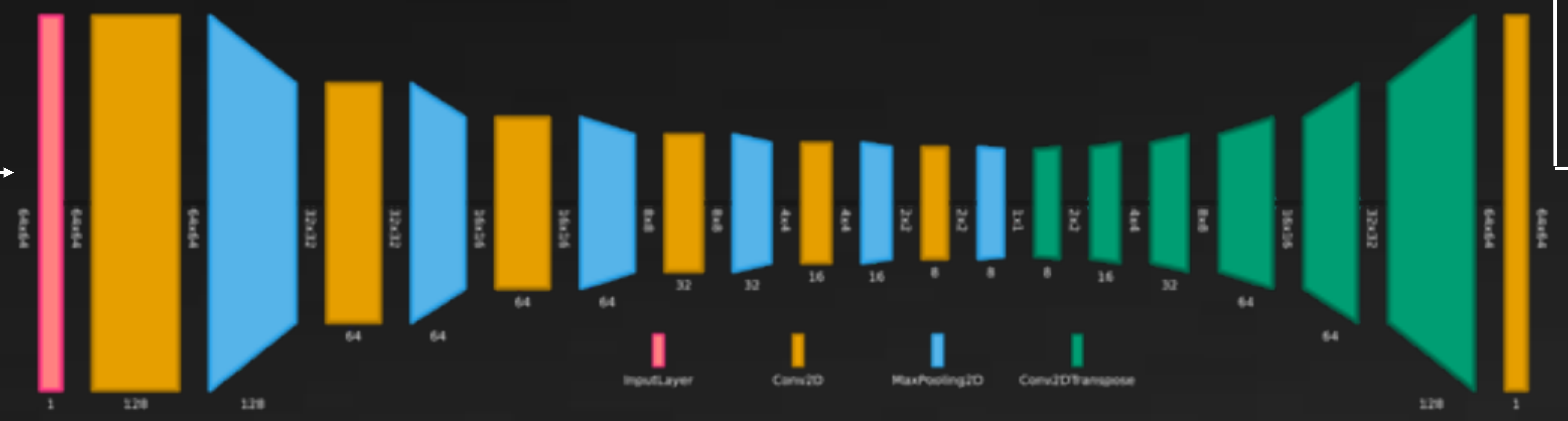
Mean of initial guess

Initial guess – mean = Δ_{init}

Min-max normalization(Δ_{init})

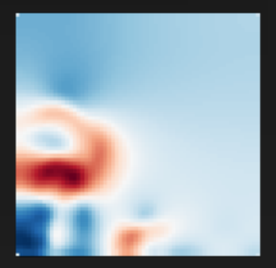
Δ_{init}





Converged solution – initial guess=Δ_{sol}

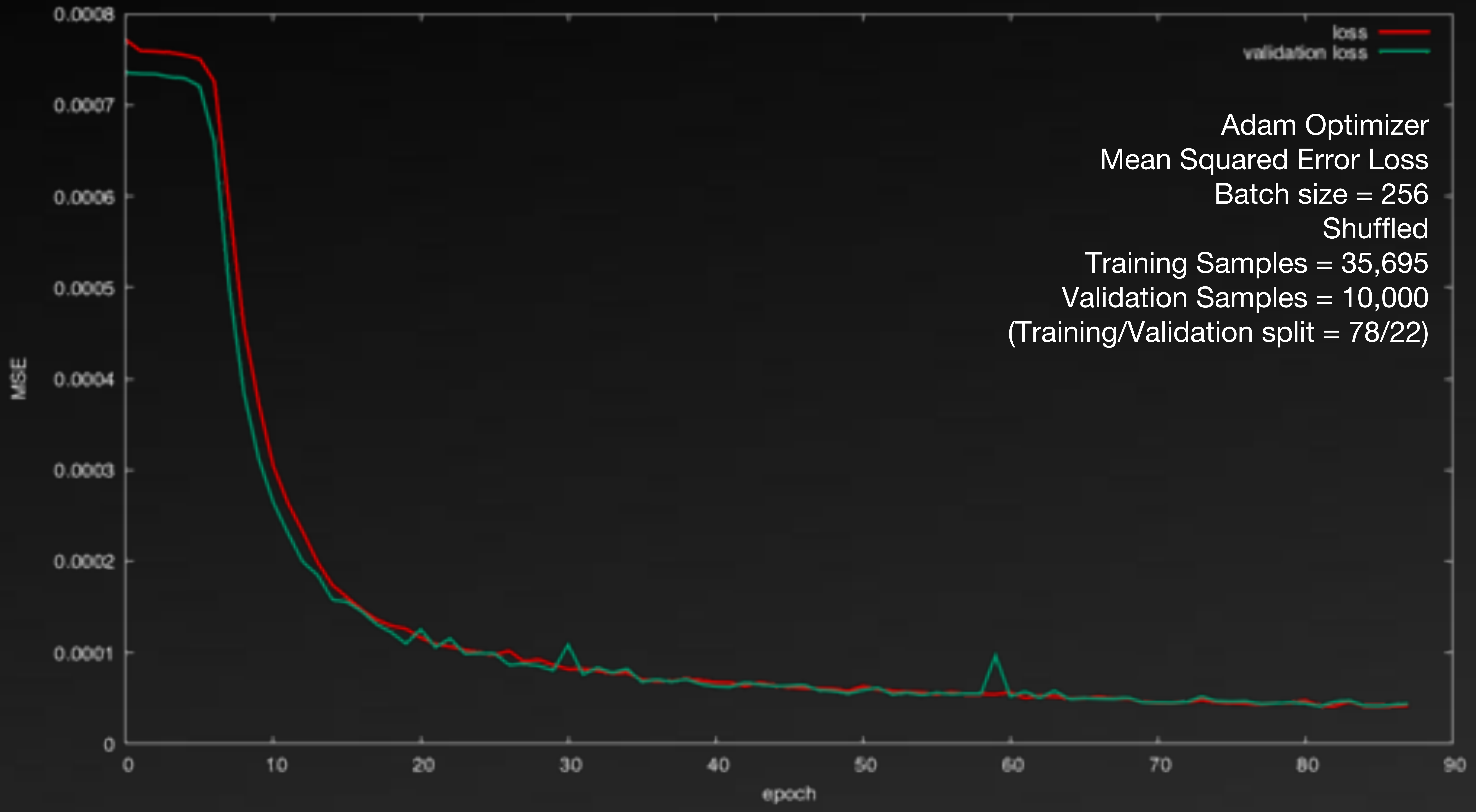
Min-max normalization(Δ_{sol})



Δ_{sol}

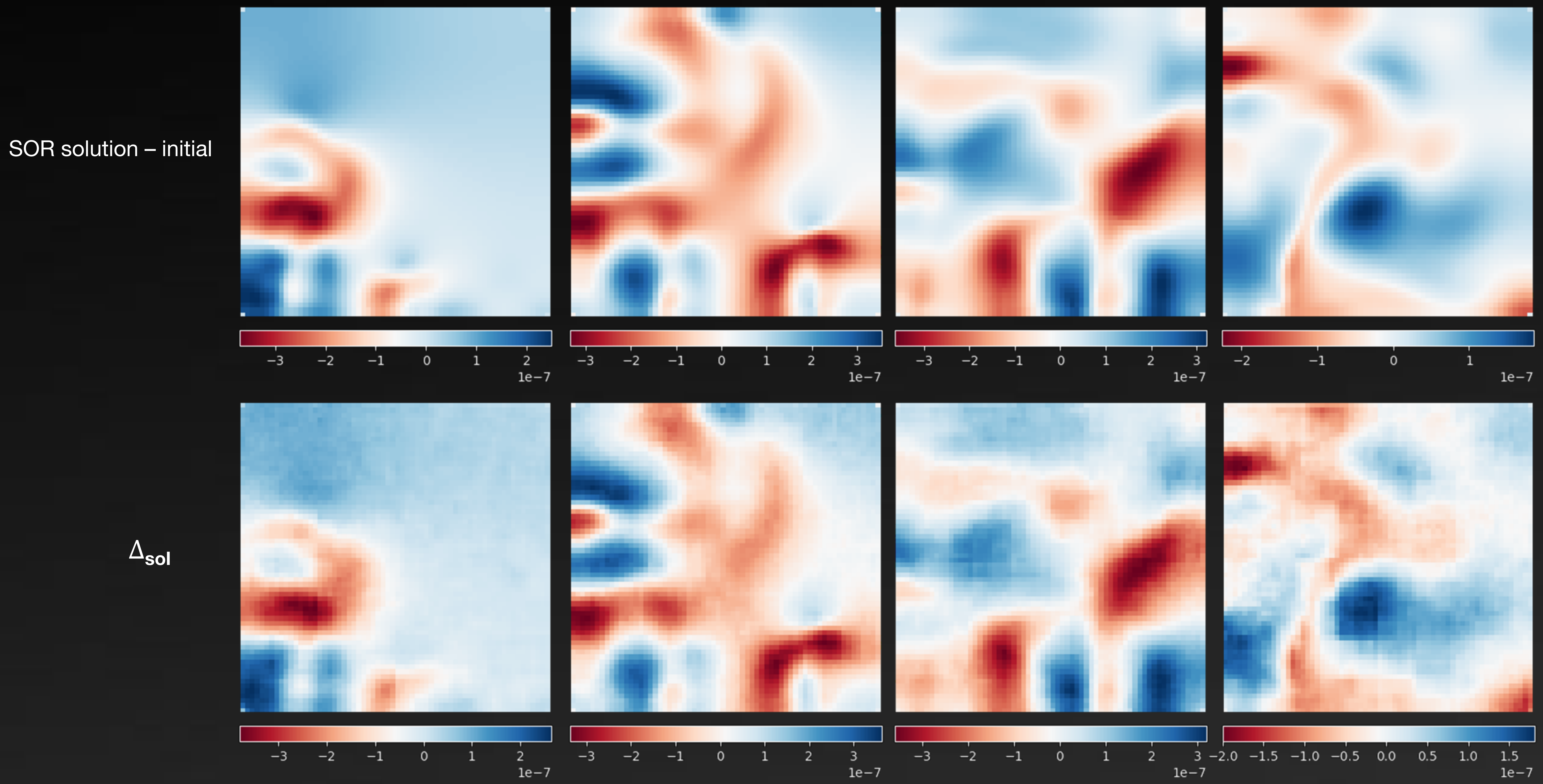
‘improved’ initial guess = initial guess + Δ_{sol}

ML model Autoencoder network



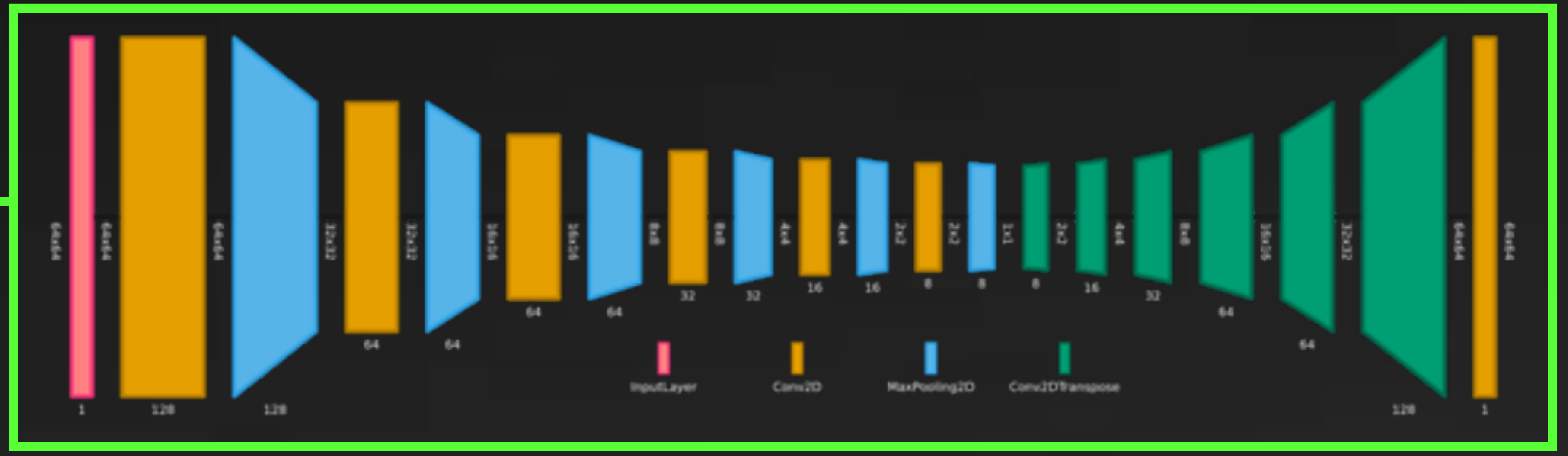
ML model

Autoencoder Output & SOR (Ra = $5x10^6$)

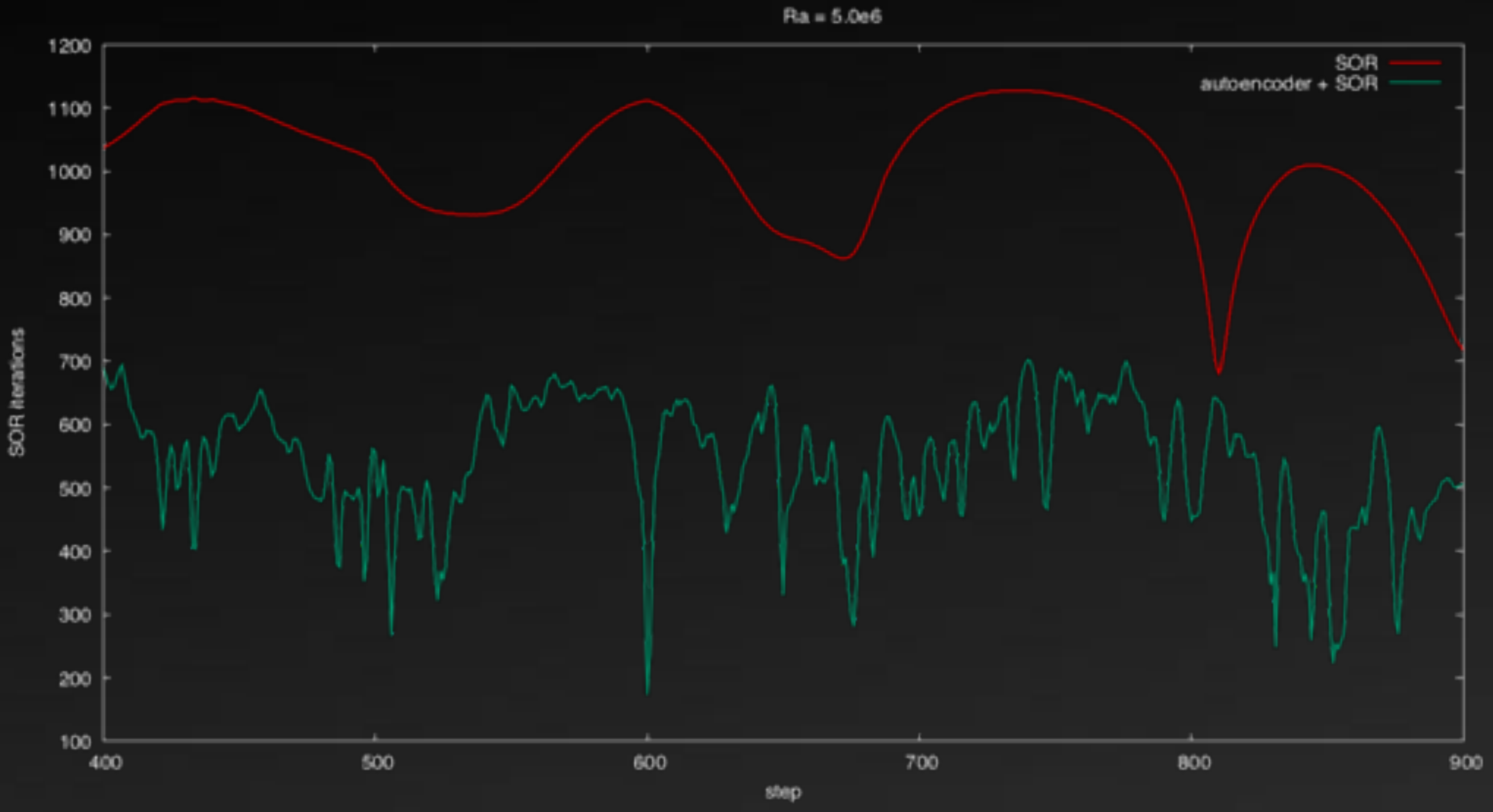


Algorithm

Set t=0, n=0
Initialise variables u,v,p,T
While t<tend
- choose delta t (timestep)
- set boundary conditions for u, v, T
- compute T^{n+1}
- compute F^n and G^n
- compute RHS of pressure equation
- set it = 0
- set p = autoecoder prediction
- while $it < it_{max}$ and (residual norm > tolerance)
  - perform SOR cycle
  - compute residual norm of pressure equation
  - it += 1
- compute u^{n+1} and v^{n+1}
- t += delta t
- n +=1

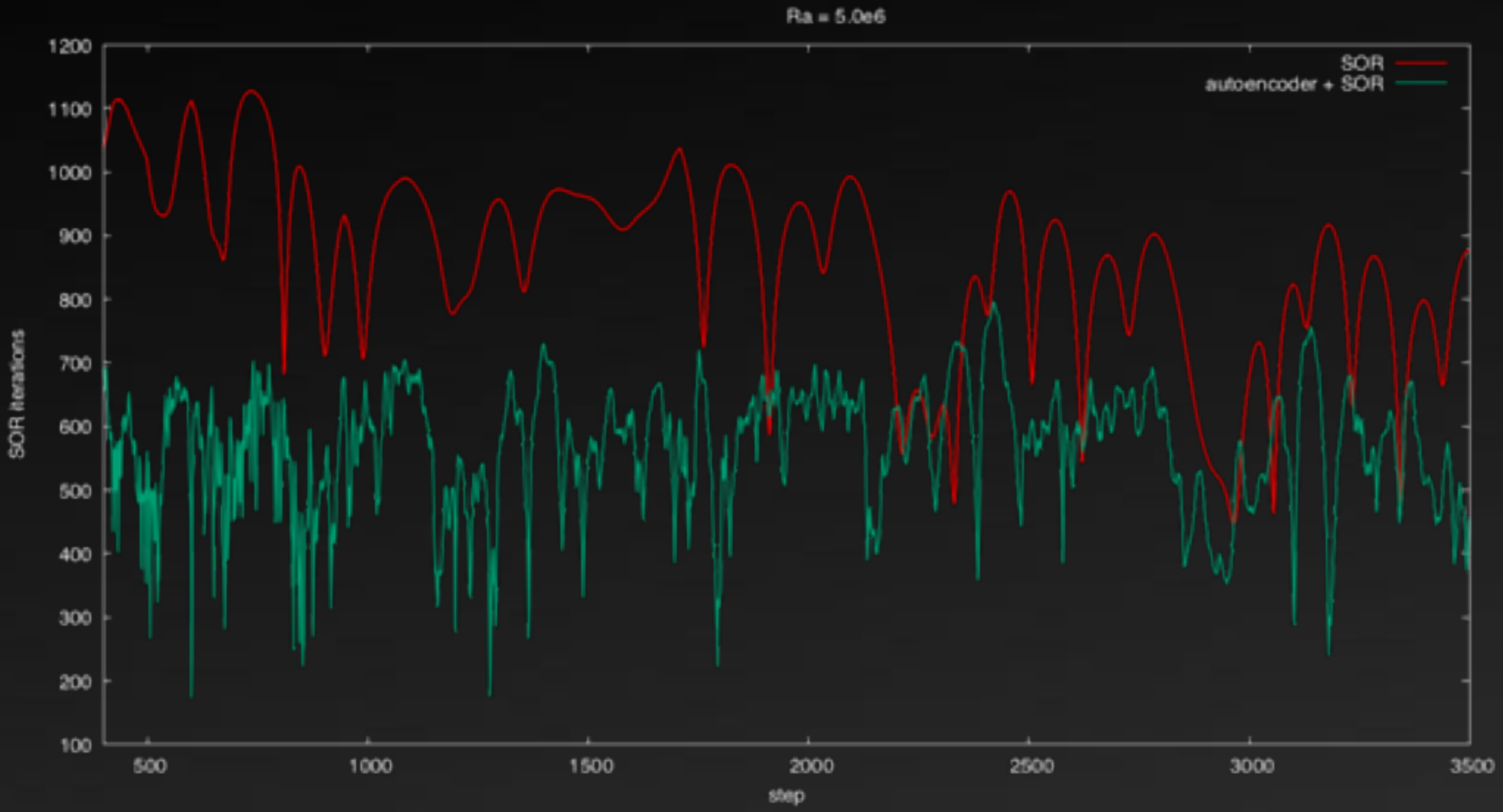


ML model



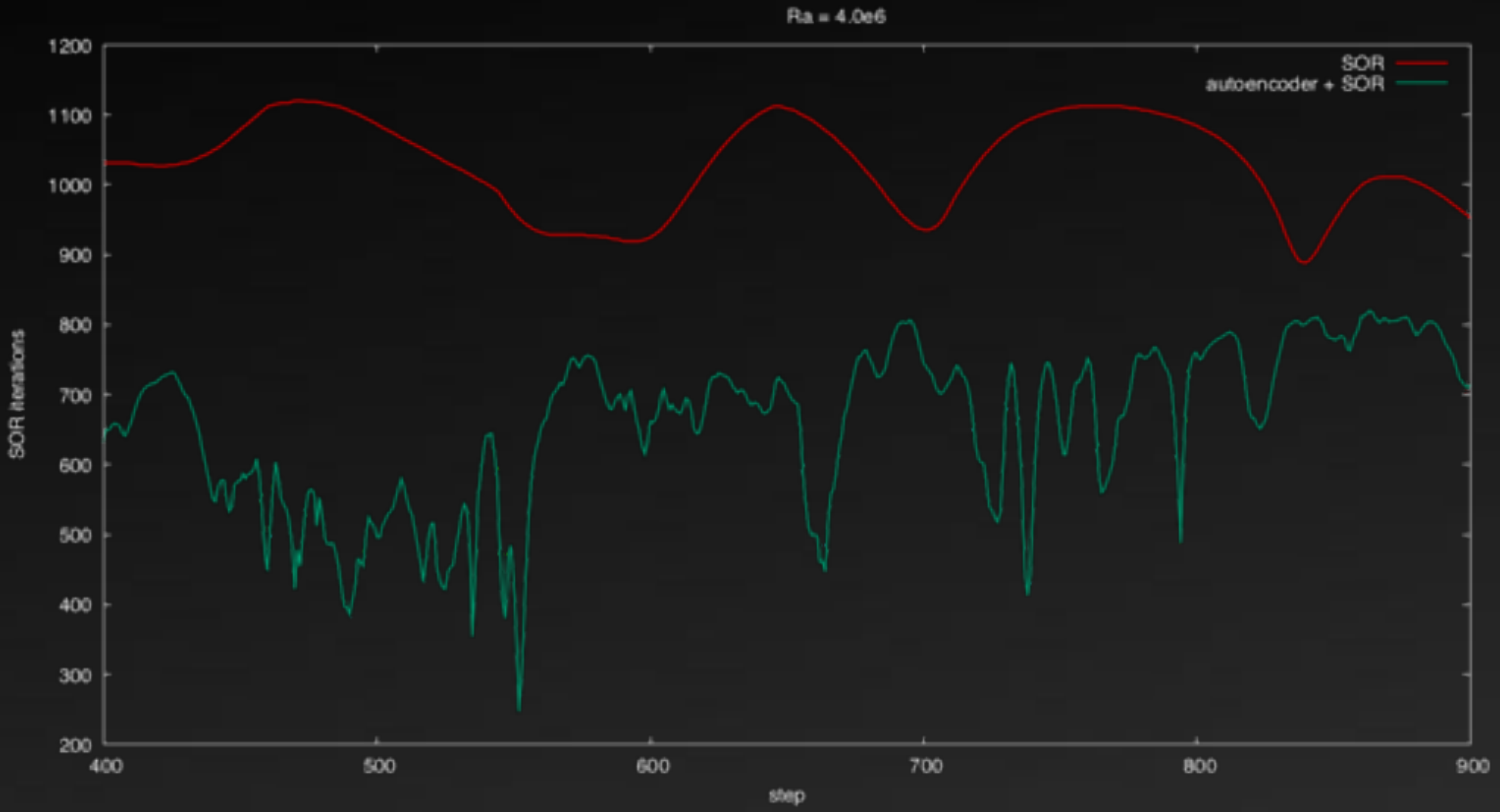
Ra = 5.0e6
SOR
autoencoder + SOR
SOR iterations
100
200
300
400
500
600
700
800
900
1000
1100
1200
400
500
600
700
800
900
step

ML model



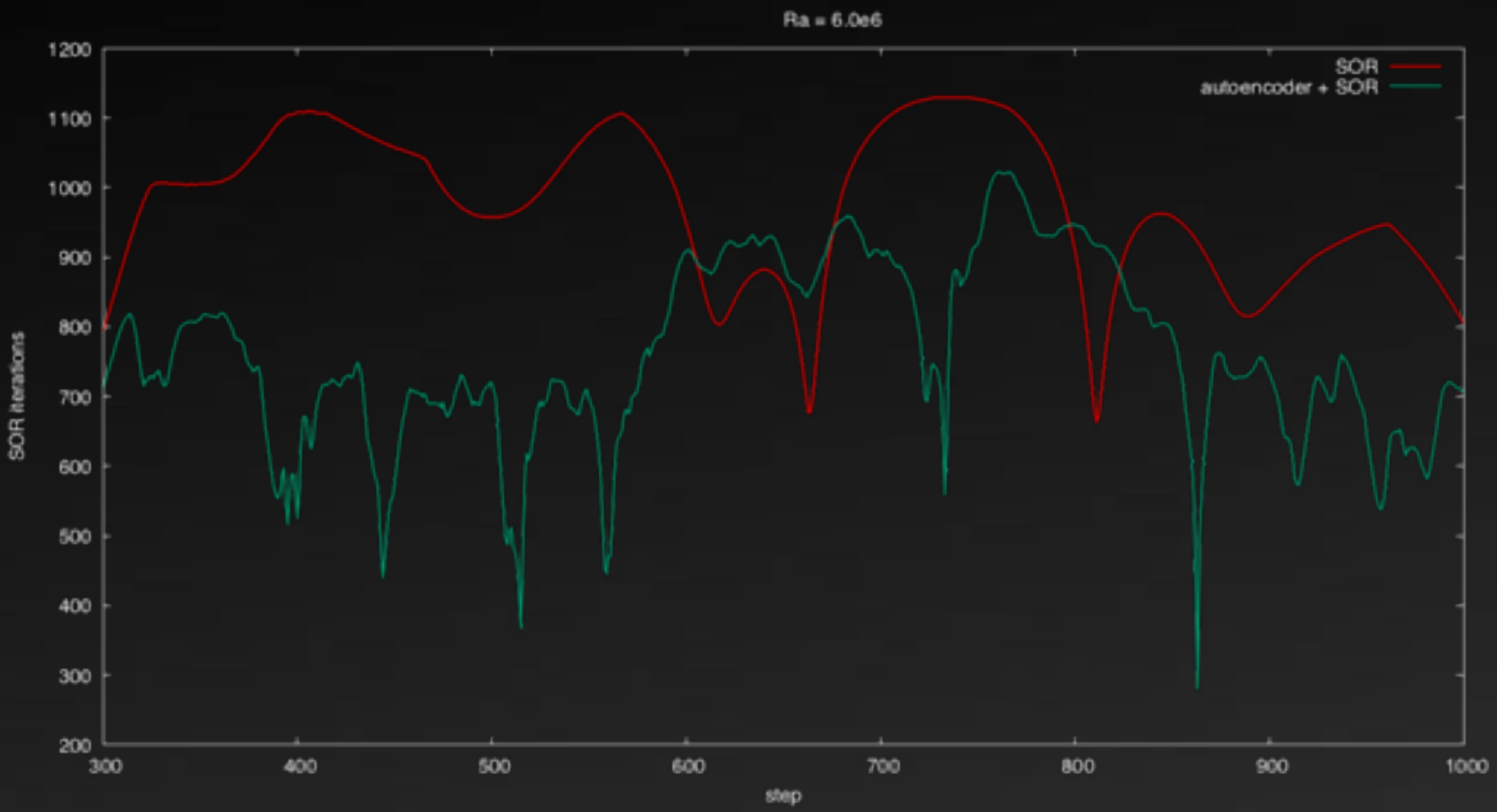
Ra = 5.0e6
SOR
autoencoder + SOR
SOR iterations
step
1200
1100
1000
900
800
700
600
500
400
300
200
100
500
1000
1500
2000
2500
3000
3500

ML model



Ra = 4.0e6
SOR
autoencoder + SOR
SOR iterations
step
1200
1100
1000
900
800
700
600
500
400
300
200
400
500
600
700
800
900

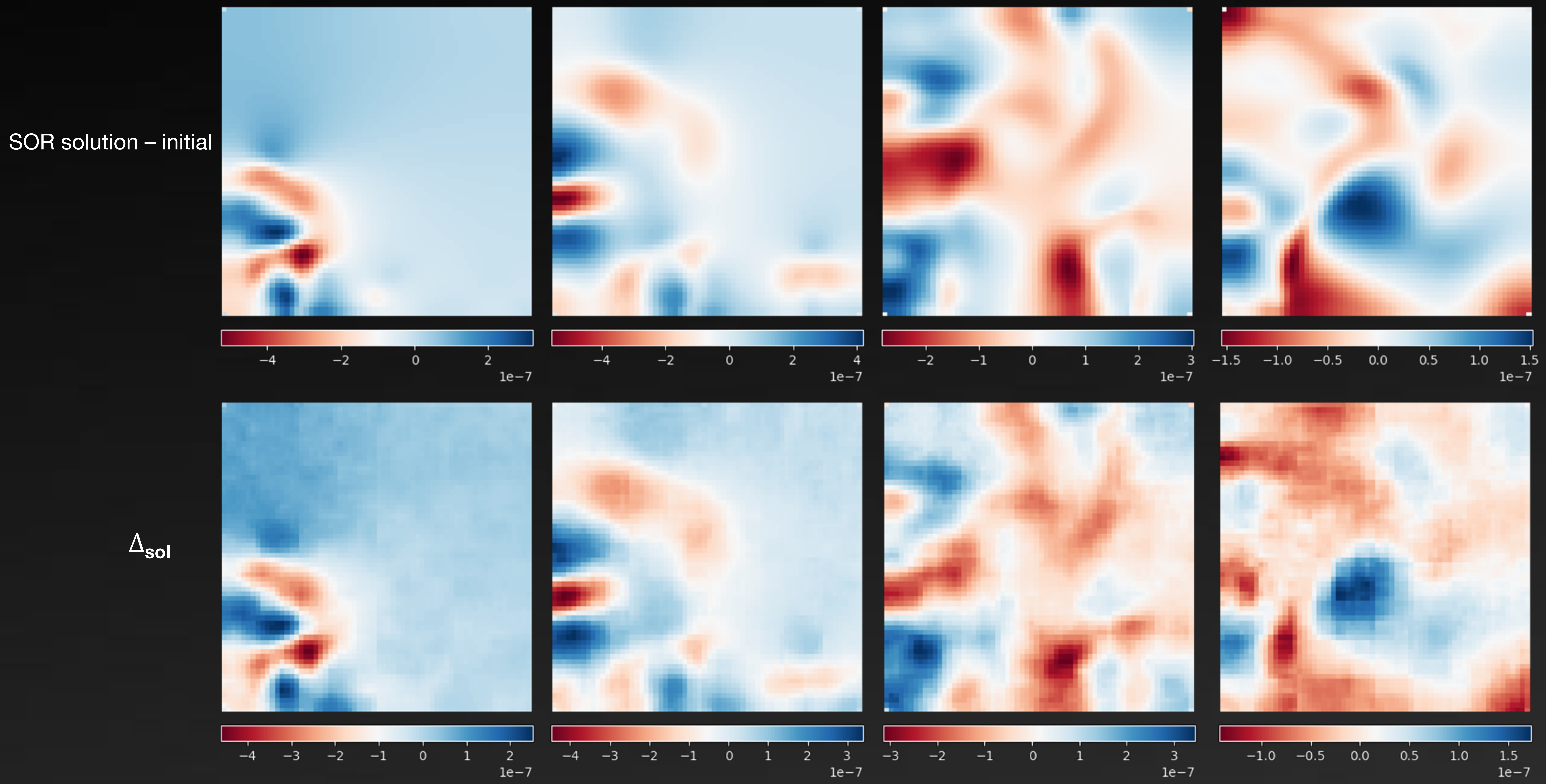
ML model



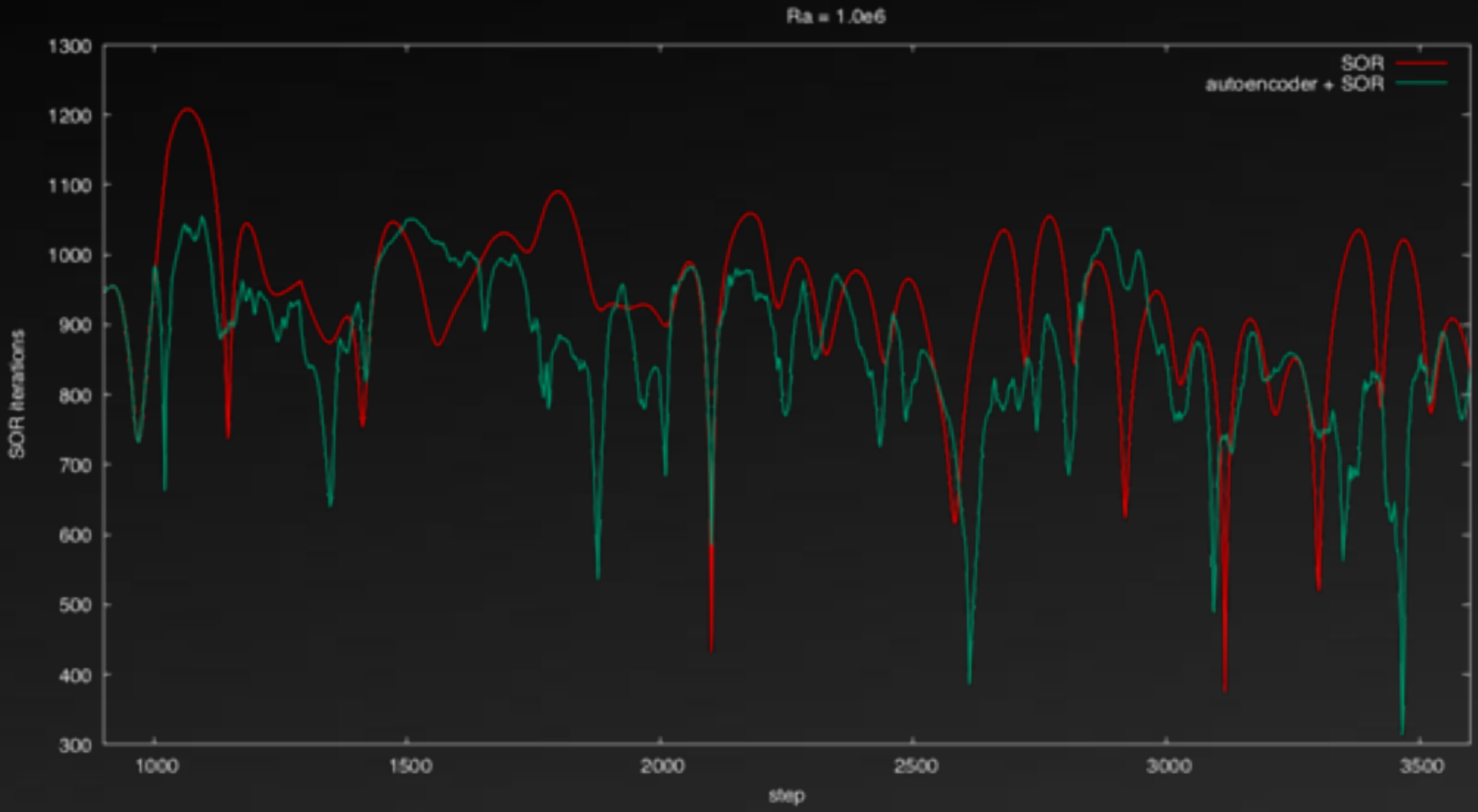
Ra = 6.0e6
SOR
autoencoder + SOR
SOR iterations
step
1200
1100
1000
900
800
700
600
500
400
300
200
300
400
500
600
700
800
900
1000

ML model

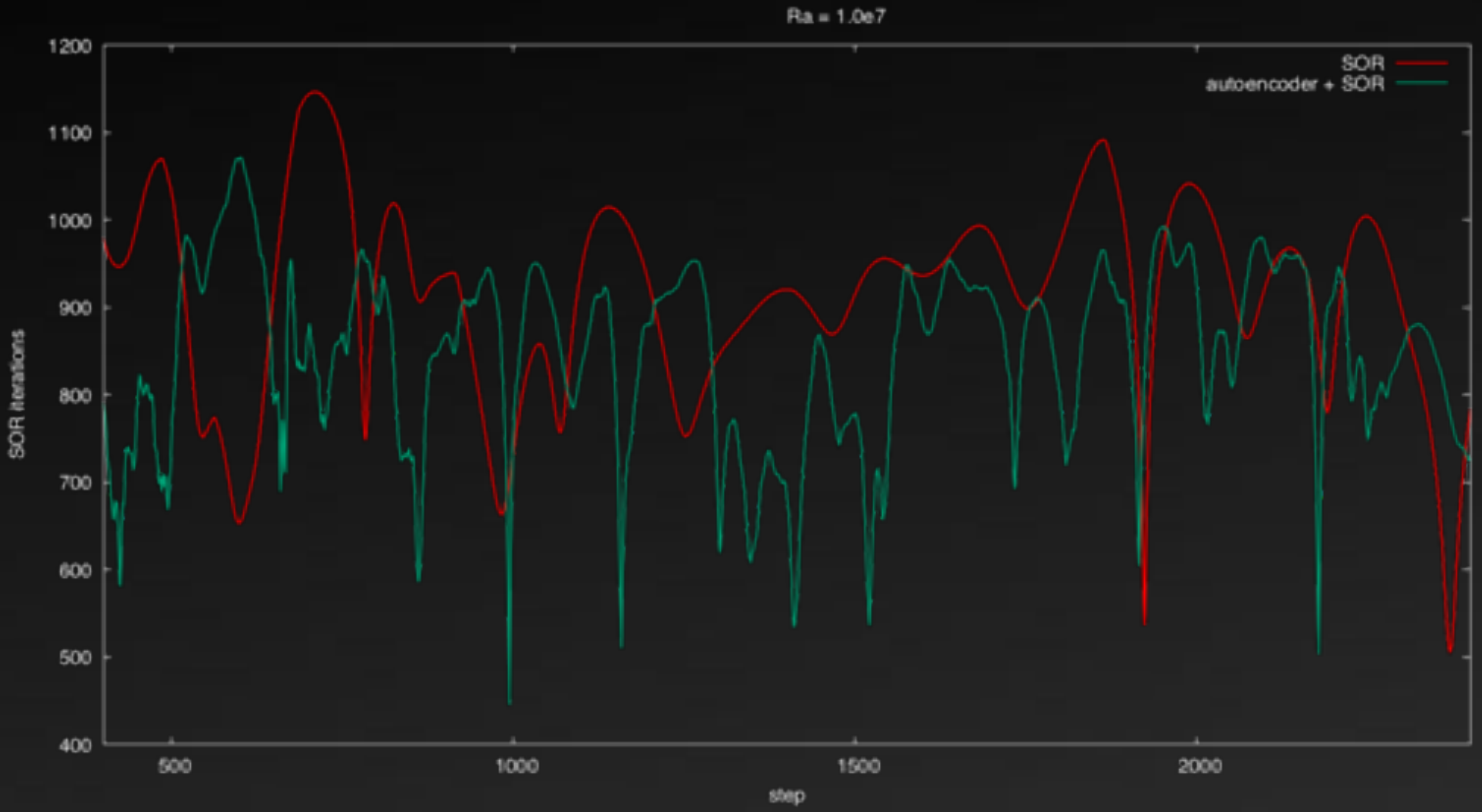
Autoencoder Output & SOR (Ra =6x10^6)



ML model


Ra = 1.0e6
SOR
autoencoder + SOR
SOR iterations
step
1300
1200
1100
1000
900
800
700
600
500
400
300
1000
1500
2000
2500
3000
3500

ML model



Ra = 1.0e7
SOR
autoencoder + SOR
SOR iterations
step
1200
1100
1000
900
800
700
600
500
400
500
1000
1500
2000

Thank You

Justin Freeman
justin.freeman@bom.gov.au