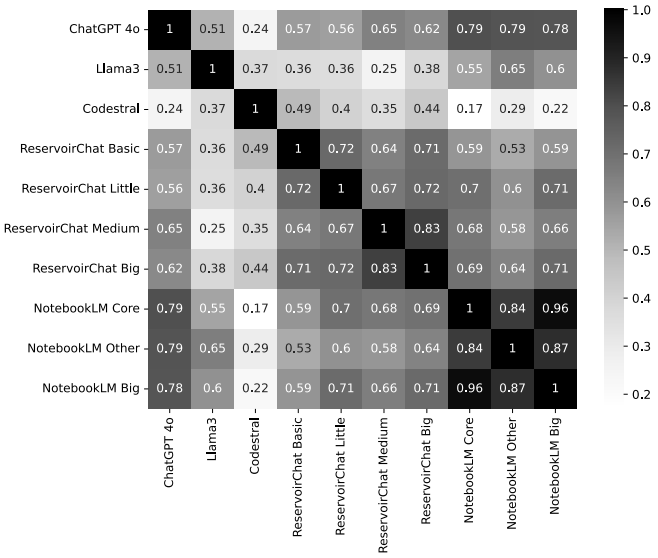
APPENDIX

A. Pearson Correlation between models for the whole questions

Fig. 11. Pearson correlation matrix of every model for the whole questions



For each model, we concatenated the 3 answers they made for all questions of the benchmark into one vector, and we computed the Pearson correlation between all of them, giving us the Heatmat of correlation that we can see in Figure 11.

B. Similarity Rate

Fig. 12. Similarity Rate Table between the answers of ReservoirChat$_{Big}$ and ChatGPT-4o, Llama3, Codestral and NotebookLM$_{Big}$

%	Knowledge	Code
Mean	81,5	62,8
Median	75,0	66,8

Big ReservoirChat outputs answers significantly similar to the other models, with a correlation rate ranging from 62.8% to 81.5%, with coherent medians, not generating any remarkable aberrant error that no other models generated. Therefore, Even though, it has overall best performance than the other models, it seems to give the same good answers and errors than the others. For each question, the mean of the 4 other models answers were divided by the ReservoirChat$_{Big}$ score. Then, each value were put into a series to calculate its mean and median.

C. ReservoirChat: QA benchmark

1) Knowledge questions: Important: there is only one unique correct answer for each question.

a) Beginner:

b) 1. What is an echo state network?:

- A. A place where anything is kept in store.
- B. A recurrent neural network in which usually only the output neurons are trained.
- C. A black box component that receives an input signal to be read out and mapped by another process.
- D. A large natural or artificial lake used as a source of water supply.

Correct: B

c) 2. What is reservoir computing?:

- A. A machine learning paradigm for timeseries.
- B. The construction of a computer model of a petroleum reservoir.
- C. A branch of fluid dynamics that studies the design of tank trucks to reduce rollover accidents.
- D. It is only a computational framework in which all the computations are done by a pool of water.

d) 3. What is call the readout? (in the context of reservoir computing):

- A. A linear regression that models the Markovian process of a support vector machine.
- B. An efficient way for input processing in order to read only part of the inputs.
- C. A black box component that receives an input signal to be read out and mapped back to the input for predictive coding.
- D. A trainable layer that extracts the dynamics of a reservoir to produce some desired outputs.

Correct: D

e) 4. On which task is reservoir computing known to compete with the state-of-the-art?:

- A. Video segmentation.
- B. Natural language processing.
- C. Image recognition.
- D. Chaotic timeseries prediction.

Correct: D

f) 5. On which task is it challenging to apply reservoir computing compared to other state-of-the-art methods?:

- A. A task with small univariate dataset.
- B. Video segmentation.
- C. Chaotic timeseries prediction.
- D. Classification of timeseries with few classes.

Correct: B

g) 6. Approximately how many neurons are used inside an echo state network?:

- A. Around 1 thousand neurons.
- B. Around 100 thousand neurons.
- C. Around 10 neurons.
- D. Around 1 million neurons.

Correct: A

h) 7. What is the purpose of using ridge regression instead of linear regression for the readout?:

- A. Reduce the computational cost.
- B. Improve numerical stability.

- C. Improve explainability of the model.
- D. Avoid the exploding/vanishing gradient problem.

Correct: B

i) 8. How are the weights most often initialized in an echo state network?:

- A. They can be randomly initialized and then scaled to have a specific spectral radius.
- B. They are tuned according to an autocorrelation Hebbian learning rule.
- C. Trained using a linear regression.
- D. Each neuron is connected to only one input.

Correct: A

j) Intermediate:

k) 9. What is the difference between 'echo state network' and 'reservoir computing'?:

- A. Reservoir computing is a type of recurrent neural network model based on the philosophy of the echo state network.
- B. There is no difference, we can use the terms reservoir computing and echo state network indistinctly.
- C. An echo state network is a type of recurrent neural network model based on the reservoir computing paradigm.
- D. In reservoir computing, the reservoir part is a physical system, in echo state networks, the reservoir part is a recurrent neural network.

Correct: C

l) 10. Are there other forms of reservoir computing than echo state networks?:

- A. No, an echo state network is not even a form of reservoir computing.
- B. Yes, any random kernel method, even those who don't apply to timeseries, are considered to be a form of reservoir computing.
- C. No, reservoir computing only refers to the philosophy behind echo state networks.
- D. Yes, there are for instance physical reservoirs or liquid state machines, in which the reservoir is a spiking neural network.

Correct: D

m) 11. What is a liquid state machine?:

- A. An architecture in which the reservoir part is a pool of spiking neurons.
- B. A physical reservoir in which the reservoir is a reservoir of liquid, usually water.
- C. Liquid state machine and reservoir computing designate the same concept.
- D. A computational model of the hippocampus using the reservoir computing paradigm.

Correct: A

n) 12. Why is it called 'computing at the edge of chaos'?:

- A. Because it is common to add noise (and thus chaos) to the reservoir in order to stabilize its activity.
- B. Because reservoir computing works best for chaotic timeseries forecasting.
- C. Because reservoir computing often works best when the dynamics of the reservoir are approaching chaotic dynamics, but are not chaotic.
- D. It requires computers to be physically placed at the edge of a chaotic environment, such as a volcano or a kindergarten.

Correct: C

o) 13. What is the 'echo state property'?:

- A. The echo state property allows the reservoir to operate with chaotic behavior to enhance computational power.
- B. The ability to memorize past activity and cycle over it (the echo).
- C. The ability to perfectly reconstruct any input signal from the reservoir activity.
- D. The influence of initial states on the reservoir dynamics decays over time, ensuring that the current state primarily reflects recent inputs.

Correct: D

p) 14. What are some of the most important hyperparameters?:

- A. The spectral radius of the recurrent weight matrix, the leak rate, the input scaling.
- B. The spectral radius of the recurrent weight matrix, the reservoir connectivity, the weight distribution.
- C. The spectral radius of the recurrent weight matrix, the reservoir connectivity, the encoding of the input.
- D. The activation function, the reservoir connectivity, the regularization parameter.

Correct: A

q) Advanced:

r) 15. How explainable are reservoir computing models?:

- A. Reservoir computing models are generally less explainable due to their reliance on complex, nonlinear dynamics within the reservoir, making it difficult to trace the exact path of information processing.
- B. Reservoir computing models are fully explainable because they rely on predefined, static patterns that do not change over time.
- C. Reservoir computing models are highly explainable because they use simple linear transformations that can be easily interpreted.
- D. Reservoir computing models are completely explainable due to their deterministic nature, which allows for perfect traceability of every computation.

Correct: A

s) 16. To what extent do the results vary between two differently initialized reservoirs?:

- A. The results between two differently initialized reservoirs are completely unpredictable and random, regardless of the input data.
- B. The results between two differently initialized reservoirs are always identical because the initialization has no impact on the reservoir's dynamics.

- C. The results between two differently initialized reservoirs can vary somewhat, but the overall performance and behavior of the reservoir are generally robust to different initialisations, provided the reservoir is sufficiently large and well-designed.
- D. The results between two differently initialized reservoirs vary significantly because the initialization determines the exact sequence of outputs.

Correct: C

t) NEW Knowledge Based:

u) 17. What is the effective spectral radius?:

- A. The real spectral radius of the matrix W, that is always a bit different from the specified spectral radius.
- B. A weighted sum of all eigenvalues norms, that takes into account the distribution of the spectrum.
- C. A value that has similar properties to the spectral radius of a matrix, taking into account the full reservoir equation.
- D. The norm of the singular values for each neuron. It is a way to evaluate the impact of each neuron on the reservoir.

Correct: C

v) 18. What is a deep reservoir?:

- A. A deep reservoir is a reservoir computing architecture that consists of multiple layers of interconnected reservoirs, allowing for hierarchical processing and the capture of more complex temporal dynamics.
- B. An underground gas or petroleum reservoir that cannot be reached using traditional tools and infrastructure.
- C. A deep reservoir is a reservoir that employs deep learning techniques, such as backpropagation, to train the weights within the reservoir, enhancing its ability to learn from data.
- D. A deep reservoir is a reservoir that uses extremely large and dense weight matrices to store vast amounts of data.

Correct: A

w) 19. What is the use of an orthogonal matrix in the reservoir equation?:

- A. An orthogonal matrix can be represented in a condensed form, improving matrix multiplication computation time.
- B. An orthogonal matrix in the reservoir equation is used to prevent any interaction between neurons, maintaining independence.
- C. This is a trick question, there is no point in using an orthogonal matrix.
- D. It augments the memory capacity of the reservoir.

Correct: D

x) 20. What is a Conceptor?:

- A. A conceptor is a mathematical function used to compress the data within a reservoir, reducing its dimensionality for faster processing.
- B. A conceptor is a hardware component that accelerates the computation of reservoir dynamics by offloading calculations to a dedicated processor.
- C. A derivation of reservoir computing which can store and recall patterns.
- D. A conceptor is a specialized type of neuron within a reservoir that is designed to store and retrieve specific concepts or patterns.

Correct: C

2) Code questions:

a) Code (not debug) questions:

b) 1. I want to train my echo state network on multiple timeseries that have different lengths. I have seen in the documentation that you can put a 3D numpy array with shape (timeseries, timesteps, dimensions), but it wouldn't work in my case as the timeseries have different lengths.:

- A. There is no way to do that in ReservoirPy as it is most probably not a good idea to train a model with different lengths and it would induce unexpected results.
- B. You can pass a list of 2D numpy arrays that represents timeseries. As lists can contain numpy arrays of different shapes, you can specify timeseries with different lengths.
- C. You would have to pad every timeseries with zeros and then concatenate them.
- D. As NumPy doesn't support sparse arrays, it is best to use xarray for this use-case.

Correct: B

c) 2. Make me a reservoir, with 1000 neurons, and with a uniform distribution of weights, and a sparsity of 95%.:

- A.

```
from reservoirpy as rpy
reservoir = rpy.nodes.Reservoir(neurons=1
    _000, connectivity=0.05, weights="
    uniform")
```

- B.

```
from reservoirpy as rpy
reservoir = rpy.nodes.Reservoir(units=1
    _000, sparsity=0.95, W=rpy.mat_gen.
    uniform)
```

- C.

```
from reservoirpy as rpy
reservoir = rpy.Reservoir(units=1_000,
    rc_connectivity=0.05, distr="uniform")
```

- D.

```
from reservoirpy as rpy
reservoir = rpy.nodes.Reservoir(units=1
    _000, rc_connectivity=0.05, W=rpy.
    mat_gen.uniform)
```

Correct: D

d) 3. Create a model in which there are several reservoirs connected in a chain, and a readout at the end.:

- A.

```
from reservoirpy.nodes import Reservoir,
    Ridge
model = [Reservoir(100, name="1"),
    Reservoir(100, name="2"),
          Reservoir(100, name="3"),
              Reservoir(100, name="4"),
          Reservoir(100, name="5"), Ridge(
              ridge=1e-5)]
```

- B.

```
from reservoirpy.nodes import Reservoir,
    Ridge
model = Reservoir(100, name="1") >>
    Reservoir(100, name="2")
         >> Reservoir(100, name="3") >>
             Reservoir(100, name="4")
         >> Reservoir(100, name="5") >>
             Ridge(ridge=1e-5)
```

- C.

```
from reservoirpy.nodes import Reservoir,
    Ridge
model = Reservoir(100) > Reservoir(100) >
    Reservoir(100)
         > Reservoir(100) > Reservoir(100)
             > Ridge(ridge=1e-5)
```

- D.

```
from reservoirpy.nodes import Reservoir,
    Ridge
model = Ridge(ridge=1e-5, previous=
    Reservoir(100, name="5",
          previous=Reservoir(100, name="4",
          previous=Reservoir(100, name="3",
          previous=Reservoir(100, name="2",
          previous=Reservoir(100, name="1")
              )))))
```

Correct: B

e) 4. Write me an echo state network that can efficiently use the many CPU cores my machine has.:

- A.

```
import reservoirpy as rpy
rpy.set_param("backend", "parallel")

from reservoirpy.nodes import ESN
model = ESN(units=100)
model.fit(train_data, train_data)
```

- B.

```
from reservoirpy.utils import parallel
from reservoirpy.nodes import ESN
model = ESN(units=100)
with parallel(n_jobs=-1):
    model.fit(train_data, train_data)
```

- C.

```
from reservoirpy.nodes import ESN
model = ESN(units=100, workers=-1)
model.fit(train_data, train_data)
```

- D. ReservoirPy already parallelizes computation by default to ensure the best performance.

Correct: C

f) 5. I have a model with several trainable readouts inside as such::

```
from reservoirpy.nodes import Reservoir,
    Ridge

model = Reservoir(100, name="R1") >> Ridge
    (name="readout1")
model >>= Reservoir(100, name="R2") >>
    Ridge(name="readout2")
model >>= Reservoir(100, name="R3") >>
    Ridge(name="readout3")
```

g) How can I fit the model, by specifying the Y values to each Ridge node?:

- A. It is not possible to do such a thing in ReservoirPy as it wouldn't make sense.
- B. You can pass a dict as a y parameter: 'model.fit(X, "readout1": Y1, "readout2": Y2, "readout3": Y3,)'.
- C. You would have to fit each part separately before concatenating them.
- D. You can specify the node names as parameters and ReservoirPy will dispatch them correctly: 'model.fit(X, readout1=Y1, readout2=Y2, readout3=Y3)'.

Correct: B

h) 6. I have a NumPy array `X` of shape `(timeseries, timesteps, dimensions)` and I want to classify them according to my `Y` array of shape `(timeseries, )` which contains numbers from 0 to 9 according to the class the timeseries belongs to. How can I do that in ReservoirPy?:

- A.

```
from reservoirpy.nodes import Reservoir,
    ScikitLearnNode, Ridge
from sklearn.linear_model import
    RidgeClassifier

Y_ = Y.reshape(-1, 1, 1).repeat(X.shape
    [1], 1)

model = Reservoir(1000, lr=0.9, sr=1.0) >>
     ScikitLearnNode(RidgeClassifier,
    model_hypers=dict(alpha=1e-8))
model.fit(X, Y_)
```

- B. Reservoir computing is only a framework for time-series forecasting, it is not suited for classification.
- C.

```
from reservoirpy.nodes import Reservoir,
    ScikitLearnNode, Ridge
from sklearn.linear_model import
    RidgeClassifier

model = Reservoir(1000, lr=0.9, sr=1.0) >>
     RidgeClassifier(alpha=1e-8)
model.fit(X, Y)
```

- D.

```
from reservoirpy.nodes import Reservoir,
    ScikitLearnNode, Ridge
from sklearn.linear_model import
    RidgeClassifier

Y_ = Y.reshape(-1, 1, 1).repeat(X.shape
    [1], 1)

model = Reservoir(1000, lr=0.9, sr=1.0) >>
     ScikitLearnNode(RidgeClassifier)
model.fit(X, Y_)
```

Correct: A

i) Code Debug questions:

3) Code and Implementation Questions:

a) 7. Here is my code::

```
from reservoirpy.nodes import Reservoir,
    Ridge

model = Reservoir(units=200, lr=0.2, sr
    =1.0) >> Ridge(ridge=1e-4)

for x_series, y_series in zip(X_train,
    Y_train):
    model.fit(x_series, y_series, warmup
        =10)

y_pred = model.run(X_test[0])
```

b) Is that correct?:

- A. Calling `.fit` on a model erases the previous trained results. You can instead call `.fit` once by passing the lists `X_train` and `Y_train` as parameters.
- B. Everything is correct!
- C. `.fit` method is not suited for online training. Use `.train` instead.
- D. Reservoir parameters should be written in full form: `leak_rate`, `spectral_radius`.

Correct: A

c) 8. Here is my code::

```
from reservoirpy.nodes import Reservoir,
    Ridge

model = Reservoir(units=200, lr=0.2, sr
    =1.0, iss=0.2) >> Ridge(ridge=1e-4)

model.fit(X_train, Y_train, warmup=200)
Y_pred = model.run(X_test)
```

d) I have an error. What's wrong?:

- A. `iss` is not a parameter. For scaling the input, the correct parameter is `scale_factor`.
- B. Reservoir parameters should be written in full form: `leak_rate`, `spectral_radius`, `input_scaling`.
- C. You must first create the reservoir and readout nodes, and then connect them, in three separate lines.
- D. `iss` is not a parameter. For scaling the input, the correct parameter is `input_scaling`.

Correct: D

e) 9. Here is my code::

```
from reservoirpy.nodes import Reservoir,
    RLS

model = Reservoir(units=200, lr=0.2, sr
    =1.0) >> RLS(alpha=1e-4)

for x_series, y_series in zip(X_train,
    Y_train):
    model.fit(x_series, y_series, warmup
        =10)

y_pred = model.run(X_test[0])
```

f) I have an error. What's wrong?:

- A. The RLS node can only be trained online, but the `.fit` method is for offline training. You should use `.train` instead.
- B. The model has been trained on a list of timeseries but is run on a single timeseries.
- C. There are not enough units inside the reservoir. For this task, having at least 1000 neurons is recommended.
- D. Wrong import path: to import the `Reservoir` node, use `from reservoirpy.nodes.reservoirs import Reservoir`.

Correct: A

g) 10. Here's my code::

```
from reservoirpy.nodes import Input,
    Output, Reservoir, Ridge
R1 = Reservoir(100, lr=0.01, sr=1.)
R2 = Reservoir(100, lr=0.03, sr=1.)
R3 = Reservoir(100, lr=0.09, sr=1.)
R4 = Reservoir(100, lr=0.3, sr=1.)
R5 = Reservoir(100, lr=0.9, sr=1.)
R6 = Reservoir(100, lr=0.01, sr=1.)
readout = Ridge(ridge=1e-5, name="readout"
    )

path1, path2 = R1 >> R6, R2 >> R5
path3 = Input(name="input") >> [R1, R2, R3
    ]
path4 = R1 >> R2 >> R3 >> R4 >> R5 >> R6
    >> readout >> Output()
model = path1 & path2 & path3 & path4

model.fit({"input": X_train}, {"readout":
    Y_train}, warmup=10)
model.run({"input": X_test})
```

h) Is that correct?:

- A. The `.fit` and `.run` methods only take numpy arrays or list of numpy arrays, not dictionaries.
- B. Yes, everything is correct!
- C. There is a circular connection in the model.
- D. `path2` is not defined.

Correct: B

i) 11. Is this the correct usage of the method `partial_fit`?:

```
reservoir, readout = Reservoir(100, sr=1),
    Ridge(ridge=1e-8)

for x, y in zip(X, Y):
    states = reservoir.run(x)
    readout.partial_fit(states, y)

readout.fit()
model = reservoir >> readout
```

- A. By calling the method `.fit`, the readout forgets its previous training.
- B. The created model won't be fitted.
- C. While it works, it can be simplified by creating the model and calling `partial_fit` on it.
- D. Yes, everything is correctly coded.

Correct: D

j) 12. Here is my code::

```
reservoir, readout = Reservoir(100, sr=1),
    Ridge(ridge=1e-8)

model = reservoir >> readout

model.fit(X[:800], Y[:800], warmup=10)

steps = 1000
results = np.zeros((steps, 1))

last_output = X[800]
for i in range(steps):
    last_output = model(last_output)
    results[i] = last_output
```

k) Is that the best way to have a model that generates new values by looping on itself?:

- A. No, you can connect the readout to the reservoir in order to loop the results back as an input after training: `readout >> reservoir`.
- B. No, it won't work as the reservoir has an input dimension of 100 and the `results` array containing the results only has its feature dimension set to 1.
- C. You can call the `.autoregress(n=1000)` Model method.
- D. Yes, this is probably the best solution.

Correct: D

l) 13. Here is my code::

```
weights = np.random.choice([1, -1], p
    =[0.6, 1 - 0.6], replace=True, size
    =(200, 200))
reservoir = Reservoir(W=weights, sr=0.9,
    lr=0.6)
```

m) I created my reservoir this way, but it seems the reservoir has a very chaotic behavior, even though the spectral radius is below 1.:

- A. The rule of the spectral radius ¡1 holds for matrices with a normal distribution, not a Bernoulli one, which explains why you have a chaotic behavior with a spectral radius of only 1.
- B. The rule of the spectral radius ¡1 only holds when there is no leak rate, so that explains why you have a chaotic behavior with a spectral radius of only 1.
- C. The parameter `sr` is only valid when no weight matrix has been specified. If a matrix is already specified, this argument is ignored.
- D. The reservoir argument names are incorrect. You should use `spectral_radius` and `leak_rate`.

Correct: C

n) 14. Here's my code::

```
from reservoirpy.nodes import Input,
    Output, Reservoir, Ridge
R1 = Reservoir(100, lr=0.01, sr=1.)
R2 = Reservoir(100, lr=0.03, sr=1.)
R3 = Reservoir(100, lr=0.09, sr=1.)
R4 = Reservoir(100, lr=0.3, sr=1.)
R5 = Reservoir(100, lr=0.9, sr=1.)
R6 = Reservoir(100, lr=0.01, sr=1.)
readout = Ridge(ridge=1e-5, name="readout"
    )

path1, path2 = R1 >> R6, R2 >> R5
path3 = Input(name="input") >> [R1, R2, R3
    ]
path4 = R1 >> R2 >> R4 >> R3 >> R5 >> R6
    >> readout >> Output()
model = path1 & path2 & path3 & path4

model.fit({"input": X_train}, {"readout":
    Y_train}, warmup=10)
model.run({"input": X_test})
```

o) Is that correct?:

- A. The `.fit` and `.run` methods only take numpy arrays or list of numpy arrays, not dictionaries.
- B. Yes, everything is correct!
- C. There is a circular connection in the model.
- D. `path2` is not defined.

Correct: B