**BioAutoMATED Installation Guide**

There are two options for code installation: 1) Docker and 2) GitHub. We recommend using Docker because the package installations are automatically handled and guaranteed to work after Docker is successfully installed. However, some Macs do not run BioAutoMATED very fast on Docker, even if many resources are allocated to the Docker container. For those users, you may wish to download the packages yourself from GitHub into a conda virtual environment. If you decide to go with this installation option, please ensure that your computer or environment can install TensorFlow v1.13.1 (with or without GPUs). Unfortunately, the new Macs with M1 chips cannot install this version of TensorFlow.

**Option #1: Docker Installation** (recommended, no package installations required after Docker)

1. Install [Docker](https://docs.docker.com/get-docker/).
   1. Download the app for your specific system by following the instructions on the Docker installation page. The current instructions are reproduced in part here for ease of use. Follow the Docker instructions for your specific system.
   2. For Macs with Intel Chip: macOS must be version 10.15 or newer. That is, Catalina, Big Sur, or Monterey. We recommend upgrading to the latest version of macOS. At least 4 GB of RAM is required.
   3. For Windows with WSL 2 backend: The requirements are as follows: The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11:
      1. 64-bit processor with [Second Level Address Translation (SLAT)](https://en.wikipedia.org/wiki/Second_Level_Address_Translation)
      2. 4GB system RAM
      3. BIOS-level hardware virtualization support must be enabled in the BIOS settings.
      4. Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher. Windows 10 64-bit: Home or Pro 2004 (build 19041) or higher, or Enterprise or Education 1909 (build 18363) or higher.
      5. Enable the WSL 2 feature on Windows. For detailed instructions, refer to the [Microsoft documentation](https://docs.microsoft.com/en-us/windows/wsl/install-win10).
   4. For example, for MacOS running the Apple chip, you should double click Docker.dmg and then drag the Docker.dmg into the applications folder. Then, double click Docker.app in the Applications folder to start Docker. Docker should now be open on your computer at this point.
   5. For Linux environments, we used the following [guidelines](https://docs.docker.com/engine/install/ubuntu/) for installing Docker on a Linux instance (here, Ubuntu 18.04 LTS). Reproducing the code below for reference.
      1. sudo apt-get update
      2. sudo apt-get install \

ca-certificates \

curl \

gnupg \

lsb-release

* + 1. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
    2. echo \ "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] [https://download.docker.com/linux/ubuntu \](https://download.docker.com/linux/ubuntu%20\) $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
    3. sudo apt-get update
    4. sudo apt-get install docker-ce docker-ce-cli containerd.io
    5. apt-cache madison docker-ce (to see available versions)
    6. sudo apt-get install docker-ce=5:20.10.12~3-0~ubuntu-bionic containerd.io

1. Next, open terminal if it is not already open and navigate to a folder where you would like to work out of. For example, this may be your desktop: cd Desktop/. The next part of these instructions is adapted from the Docker installation instructions from the PyModulon package [here](https://github.com/SBRG/pymodulon).
2. Run the following command to test your Docker installation: “docker run hello-world” You should see something along the lines of “Hello from Docker! This message shows that your installation appears to be working correctly.” Do not proceed if you do not have this running successfully!
   1. Before proceeding to the next step, make sure you do not have anything running on your IP address port 8888. We will be using this port for our Jupyter notebook.
3. Finally, you are ready to download the Docker repository and get started.
   1. Pull the repository: docker pull jackievaleri/bioautomated:v2
   2. After this stage, there should be a message saying where the image was downloaded to, for example “docker.io/jackievaleri/ bioautomated:v2”. You can then start up the docker container: docker run -dp 8888:8888 docker.io/jackievaleri/bioautomated:v2 [or the relevant location]
   3. You can check what containers are running with the command docker ps -a.
   4. Now, you should be able to copy your IP address plus the port number to access the code. Your IP address is localhost if you are running this guide on your machine, or your external IP address if you are running this on a virtual machine. For example, copy localhost:8888 into your Web browser.
   5. Then, open up the 01\_BioAutoMATED\_Small\_System\_Test\_START\_HERE.ipynb. Depending on where you are, you may need to first navigate into a folder called BioAutoMATED/. You should be able to run everything.

**Option #2: GitHub Download & Conda Installation**

1. First, download the repository from GitHub at: <https://github.com/jackievaleri/BioAutoMATED> or run the following command:
   1. git clone https://github.com/jackievaleri/BioAutoMATED.git BioAutoMATED
2. If you do not already have conda, install Anaconda and add it to your path. For example,
   1. wget<http://repo.continuum.io/archive/Anaconda3-5.3.1-Linux-x86_64.sh>
   2. sudo bash Anaconda3-5.3.1-Linux-x86\_64.sh
   3. export PATH=~/anaconda3/bin:$PATH
   4. vim ~/.bashrc and add “export PATH=~/anaconda3/bin:$PATH” to the last line
   5. source ~/.bashrc or reboot
   6. Check that the install worked by trying: conda –version. There should be an output with your conda version.
3. Install Python 3.7 if you do not already have it: conda install -c anaconda python=3.7
4. Create and activate a virtual environment called automl\_py37 with python 3.7 from the environment.yml file.
   1. conda env create -f environment.yml
   2. conda activate automl\_py37
5. Finish with the last few installations that do not play well with conda:
   1. Note: you should replace instances of “pip” and “python” with “pip3” and “python3”, respectively, if you have older installations of python2 on your computer. This will force the python3 path to be used.
   2. pip install -r requirements.txt
   3. pip install autokeras==0.4.0
      1. Note, you may see this error: ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts. deepswarm 0.0.10 requires scikit-learn==0.20.3, but you have scikit-learn 0.20.2 which is incompatible.
      2. If you have this issue, you can use this instead: pip3 install autokeras==0.4.0 --no-deps
      3. Go ahead and proceed – scikit-learn incompatibilities have been factored in.
   4. apt-get install graphviz graphviz-dev -y
      1. For Mac, okay to use: brew install graphviz
      2. Note, you may see this error: Error: Failed to download resource "libpng"
      3. If you see this error, you can use pip3 install graphviz instead.
   5. pip install pygraphviz
   6. conda install -c conda-forge python-graphviz
   7. git clone <https://github.com/heuritech/convnets-keras.git>
   8. cd convnets-keras
   9. python setup.py install
   10. cd ..
   11. pip install git+<https://github.com/raghakot/keras-vis.git>
6. Deactivate the conda environment and add the environment to your ipykernel:
   1. conda deactivate
   2. python -m ipykernel install --user --name=automl\_py37
7. At this point, you should be able to use the command jupyter notebook, which should launch the Jupyter window in your web browser. Open up the notebook 01\_BioAutoMATED\_Small\_System\_Test\_START\_HERE.ipynb and select the kernel: automl\_py37.
   1. Note: Tensorflow 1.13.1 does not currently run on Macs with M1 chips.
8. If you prefer to run BioAutoMATED in command line, navigate to the folder called BioAutoMATED. Then run the following:
   1. conda activate automl\_py37
   2. python main\_classes/wrapper.py -task binary\_classification -data\_folder ./clean\_data/clean/ -data\_file small\_synthetic.csv -sequence\_type nucleic\_acid -model\_folder ./exemplars/test\_synthetic\_nucleic\_acids/models/ -output\_folder ./exemplars/test\_synthetic\_nucleic\_acids/outputs/ -verbosity 1 -input\_col seq -target\_col positive\_score -max\_runtime\_minutes 10 -num\_folds 2 -num\_final\_epochs 10 -num\_generations 5 -population\_size 5
   3. This replicates the output of the small system test described in Step 7 above. For a full list of arguments and their meaning, please reference either the .ipynb notebook or the run\_bioautomated function in the wrapper.py file.
   4. Also, please note that you will have to make the test\_synthetic\_nucleic\_acids/ folder in addition to the models/ and outputs/ subfolders if not already present.
9. To close the notebook, press Ctrl+C in terminal. All changes made to files in your current directory are saved to your local machine.

**Troubleshooting:**

**Problem:** When I pull the Docker repository, I get a “permission denied” or “unauthorized: incorrect username or password” error.

**Solution:** Login to your Docker account before pulling, for example with the command “docker login –username [your username]”.

**Problem:** Docker Desktop is running really slowly on my Mac!

**Solution:** It may be necessary to go into Docker Desktop preferences and increase the memory, swap, and disk image size. We recommend setting the CPUs to half the amount that your hardware has available, and increasing your swap and disk image sizes. We also recommend decreasing the number of file shares if possible (under File Sharing) by deleting /private, /tmp/, /var/folders.

**Problem:** The page buffers but never loads when I enter the address for the Jupyter notebook.

**Solution:** This problem could be caused by a variety of issues, but we often see it when there is an issue connecting with the port. Check that port 8888 is available on your machine. If you are using a platform like Google Cloud Platform to make a virtual machine, it is also important to [make a firewall rule](https://cloud.google.com/vpc/docs/using-firewalls) for port 8888 if you are using a virtual machine on Google Cloud Platform, for instance. For example, here is the output of the GCP VM firewall rule below. You may not want to use the 0.0.0.0/0 (open to everything) IP range if you have not specified which instances this firewall rule is applicable for, or else your system will be open to anyone.

