Simulation-based inference of gravitational waves

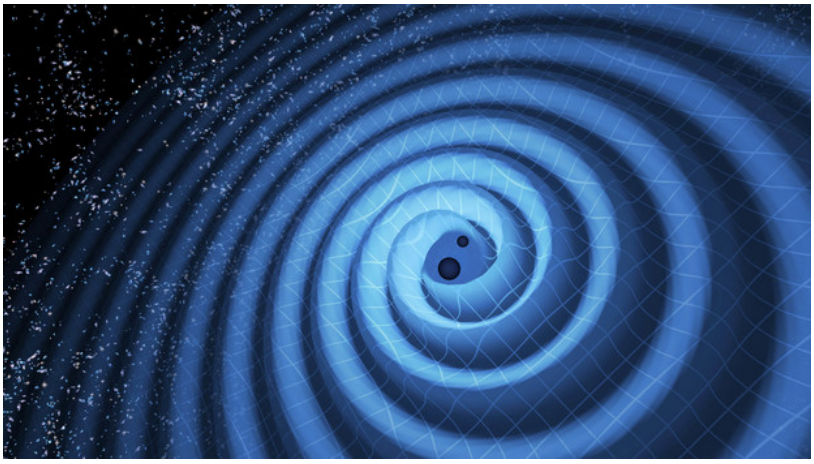
Testing performance with Vision Transfrs

submitted in partial fulfillment for the degree of master of science

Nicola Asquith
15058050

Master Information Studies
data science
Faculty of Science
University of Amsterdam

Submitted on 30.06.2024



	UvA Supervisor
Title, Name	Christoph Weniger
Affiliation	GRAPPA Institute Institute for Theoretical Physics
Email	C.Weniger@uva.nl

media/titlepage/logo-uva.png

ABSTRACT

Cover image: https://www.ligo.caltech.edu/page/what-are-gw

KEYWORDS

Simulation-Based Inference, Bayesian Inference, Gravitational Waves, UNet, Vision Transformer

GITHUB REPOSITORY

https://github.com/Nic0laA/master-thesis

1 INTRODUCTION

Gravitational Waves (GW) are ripples in the fabric of space-time originating from the acceleration of massive astronomical objects e.g. the merger of black holes or neutron stars. Analysis of gravitational waves can be used to infer properties of the source, as well as opening opportunities to observe the universe in an entirely new way.

Since the first direct detection of gravitational waves in 2015 [**?**], the detector sensitivities and survey volumes are ever-increasing. The substantial rise in detection rate of events over time is introducing significant data analysis challenges for the gravitational wave community [**?**]. For instance, current data analysis pipelines are not equipped to deal with independent signals arriving coincidentally in detectors, and scale poorly as the dimensionality of the problem increases [**?**]. This makes the analysis of large number of overlapping signals, or those containing non-stationary noise increasingly complicated and computationally expensive [**?**].

The `peregrine` inference pipeline has been developed at the UvA GRAPPA institute to help address some of these challenges [**?**]. It utilises the Simulation-based inference (SBI) method based on the TMNRE (Truncated Marginal Neural Ratio Estimation) algorithm with the U-Net Convolutional Neural Network (CNN) architecture. The `peregrine` pipeline consists of multiple rounds of network training and inference, which comes with a high simulation cost. It is therefore highly beneficial for the network to be as fast and as accurate as possible. This thesis explores the possibilities to optimise the network architecture underlying the `peregrine` code. The main research question will be:

RQ:To what extent can the optimisation of the `peregrine` *network architecture reduce the simulation budget, while still producing the same results as the original* `peregrine`*?*

Smaller sub-questions to be addressed include:

SRQ1: How can we quantify the efficiency of the underlying neural network?

SRQ2: How can more efficient sampling methods be used to further improve the computational efficiency of `peregrine`?

To explore the possibilities of improving the performance of the U-Net CNN within `peregrine`, we will consider two competing approaches - expansion and reduction. For the expansion part, we will look at two pretrained vision transformer models [**? ?**], and attention u-net [**?**]. For the reduction approach, we will look at pruning methods [**?**].

The structure of this report is as follows. Related work in section bla, background theory in section bla.

2 RELATED WORK

2.1 Gravitational wave analysis

The first direct detection of gravitational waves occurred in 2015 after the merger of two black holes [**?**]. Since then, the LIGO-Virgo collaboration has confirmed 90 gravitational wave detections from the first three observing runs. Of these 90 detections, there were 83 black hole mergers, 2 binary neutron star mergers, 3 neutron star-black hole mergers and 2 involving the merger between a black hole and a 'mystery' object that had a mass in between that of a neutron star and black hole [**?**]. The annual detection rate of events is expected to increase significantly in the fourth and fifth observing runs, increasing ∼5x each observing run [**?**]. The first direct detection of gravitational waves occurred in 2015 after the merger of two black holes [**?**]. Since then, the LIGO-Virgo collaboration has confirmed 90 gravitational wave detections from the first three observing runs. Of these 90 detections, there were 83 black hole mergers, 2 binary neutron star mergers, 3 neutron star-black hole mergers and 2 involving the merger between a black hole and a 'mystery' object that had a mass in between that of a neutron star and black hole [**?**]. The annual detection rate of events is expected to increase significantly in the fourth and fifth observing runs, increasing ∼5x each observing run [**?**].

Analysis of gravitational waves consists of two parts – detection and parameter inference [**?**]. This thesis is concerned only with the inference part. The theory of how sources emit gravitational waves is well established, and the entire gravitational waveform may be expressed by ∼15 parameters [**?**]. There are well established methods for accurately forward modelling GW waveforms, detector responses and instrument noise [**?**] e.g. the open-source Bilby code [**? ? ?**]. Therefore, given the well-known theory, and these 15 parameters as input, it is fairly straightforward to accurately simulate what you expect the GW waveform to look like. However, performing the inverse of this problem i.e. backing out the 15 parameters from a given GW waveform is significantly more challenging. Bayesian inference methods are therefore used extensively in gravitational-wave astronomy [**?**].

Due to the high dimensionality, brute-force techniques for GW parameter inference are computationally infeasible. The traditional approach involves using stochastic sampling methods [**?**], such as Markov chain Monte Carlo (MCMC) [**? ?**] or nested sampling [**?**]. However, these techniques increasingly struggle with higher dimensional data and are not feasible options in the case of overlapping signals, where one needs to now infer 30 model parameters from the data [**?**].

2.2 Swyft and Peregrine

The `Peregrine` code was developed to study broad classes of gravitational wave signals. The papers describing the development of the code [**? ?**] will form the main starting point of this thesis, and will also serve as the baseline for this new work to be benchmarked

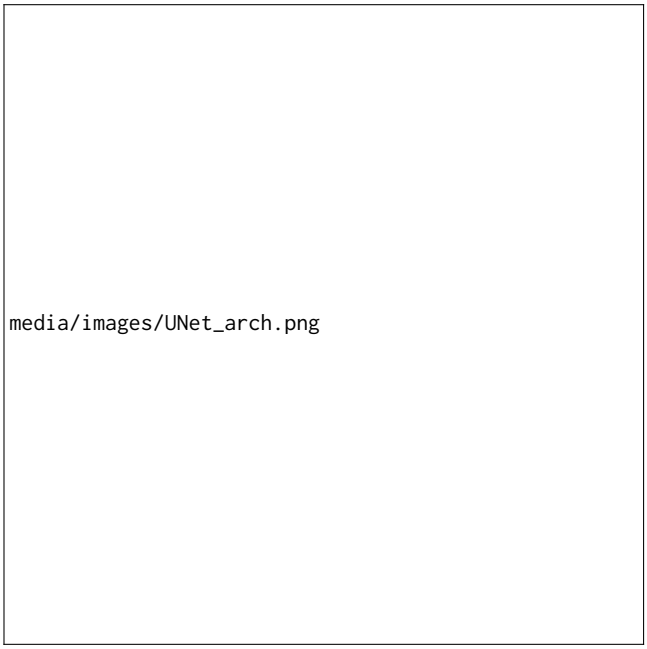


Figure 1: The U-Net architecture with a 2D image of resolution 572×572 as example. Reproduced from [?].

against. `Peregrine` implements an SBI algorithm known as Truncated Marginal Neural Ratio Estimation (TMNRE) [?] and is built on top of the `swyft` code [?]. The TMNRE algorithm estimates the marginal likelihood-to-evidence ratio, and works by training binary classifiers to distinguish jointly drawn sample pairs from marginally drawn sample pairs.

Summary statistics to MLP which calculates the log ratios.

2.3 Neural Network Architectures

2.3.1 U-Net. U-Net is a fully convolutional neural network that was originally developed in 2015 for biomedical image segmentation [?]. U-Net is efficient and fast to train, and has thus become highly popular and the gold standard for segmenting 2D medical images for which new architectures should be benchmarked against [?]. The defining feature of U-Net is its symmetrical architecture consisting of a contracting path (encoder) followed by an expansive path (decoder). In the original U-Net [?], each step in the encoding path consists of two 3×3 convolutions (unpadded in the original) doubling the number of feature channels, a ReLU and 2×2 max pooling with stride 2 for downsampling. The decoding path then follows with a 2×2 up-convolution halving the number of feature channels, concatenation with the matching feature map from the contractive path, followed by two 3×3 convolutions and ReLU operations. The architecture of U-Net, reproduced from [?], is shown in Figure 1.

2.3.2 Attention U-Net. Due to the success of the U-Net architecture, several variants have since been proposed,including U-Net++ [?] and attention U-Net [?]. U-Net++ replaces the plane skip connections in vanilla U-Net with nested dense skip connections. This is to reduce the semantic gap between the encoder and decoder, which they claim yields significant performance gains.

The Attention U-Net introduces an attention gate (AG) in order to filter the features that are passed through the skip connections. The authors claim that this increases the models sensitivity and prediction accuracy with minimal impact to the computational cost. They do this by suppressing irrelevant features and learn to only focus on the most important features.

Additive soft attention

2.3.3 Transformer models. Transformer models have proven widely succesful in the field of natural language progressing [?] and computer vision [?]. They provide a highly efficient platform for processing sequential data. Vision Transformers (ViT) have shown

Vision transformers and time series models.

The use of transformer models for 1D signals is much less widespread, but [?] have shown they were very effective at extracting features from 1D signals, specifically detecting Parkinson's disease by analysing a patients gait.

Transformers often require large amounts of training data and computing power before they are able to outperform task-specific models.

2.4 Network pruning

The implementation of attention U-Net and transformers models will increase the number of trainable parameters. This will likely place an additional overhead on the calculation time of `Peregrine` or require more training data. In order to reduce the number of model parameters pruning techniques exist such as [?].

3 BACKGROUND THEORY

3.1 Simulation-based inference

In recent years, thanks to the enormous rise in machine learning capabilities, particularly with deep neural networks, simulation-based inference (SBI) methods have experienced rapid expansion [?]. SBI is considered to be a highly simulation efficient technique and finds applications in many scientific domains including particle physics, neuroscience, epidemiology, economics, economics, climate science and astrophysics [?].

One of the major limitations of traditional MCMC and nested sampling approaches is that they require the likelihood $P(x|\theta)$, or probability of a given observation occurring to be known in advance. However, SBI does not need an explicit likelihood function up-front, because it is instead given a realistic forward simulator it can sample from.

Using Bayes' theoreom,

$$P(\theta|x) = \frac{P(\theta|x)P(\theta)}{P(x)}$$

Where $P(\theta|x)$ is the posterior of parameters θ given some observed or simulated data x, $P(\theta|x)$ is the likelihood of given data x given input parameters θ, $P(\theta)$ is the prior distribution of θ and $P(x)$ is the Bayesian evidence of x. The power of SBI arises because if you have a forward generative model, $P(x, \theta) = P(\theta|x)P(\theta)$, you are able to sample implicitly from the (simulated) likelihood [?].

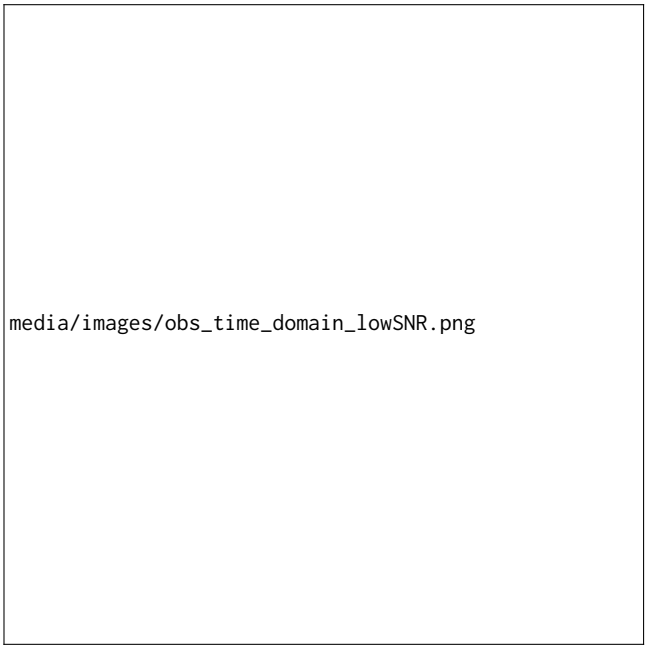


Figure 2: Example of generated gravitational wave signal in the time domain. For clarity, signals from three detectors are shown without noise. The noise signal shows the H1 signal with noise added, which is the actual signal used to train the network. The two black holes merge at the moment t=0s.

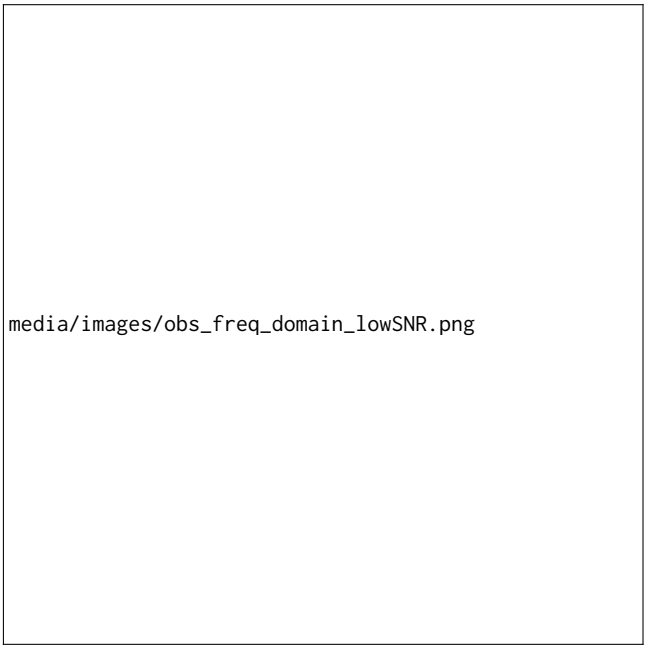


Figure 3: Example of generated gravitational wave signal in the frequency domain. For clarity, signals from three detectors are shown without noise. The noise signal shows the H1 signal with noise added, which is the actual signal used to train the network. The two black holes merge at the moment t=0s.

3.2 TMNRE

4 METHODOLOGY

Very much still a work in progress

4.1 Description of the data

The waveform data consists of 1D signals in both time and frequency domains, collected from three separate detectors (referred to as H1, L1, V1) that have captured the event simultaneously. The signal in the time domain consists of three channels (three detectors) and 8192 data points corresponding to 4 s of collection time at a sampling frequency of 2048 Hz. An example of the signal in the time domain is shown in Figure 2. The time domain can be transformed to the frequency domain through the Fourier transform. The frequency domain contains 4197 data points and six channels (real and imaginary parts of the time signal from three detectors). An example of the signal in the frequency domain is shown in Figure 3. Both the time and frequency domains are fed into the neural network, since they have different information about the GW encoded [**?**].

4.2 Generation of the waveforms

All of the waveforms used in this study were generated using the open-source Bilby code [**?**]. The Bilby code has functionality to

which was controlled using the `swyft` library [**?**], which is highly efficient and allows the user to implement their own `Simulator` class [**?**].

The waveform data consists of 1D signals

Gravitational waves are detected using highly sensitive laser interferometers.

The gravitational waveforms of three LIGO detectors (Hanford, Washington and Livingston) were generated as a function of 15 independent parameters. Ten of these parameters represented intrinsic properties of the source e.g. the masses and spins of the two black holes, while five of these are extrinsic parameters e.g. the distance from the source and orientation in the sky with respect to the observations. Further details of the parameters are listed in Table 1. Simulating the waveforms for training the neural network is necessary since we need ~100000 waveforms for training and experimentally measured signals are rare (only 90 detections so far).

Only low SNR case considered because high SNR is unrealistic and it is important for the network to determine signal from noise. Peregrine benchmark.

Complete table

The data used in this study was simulated using the open-source Bilby code [**?**]. Only low signal to noise signal will be considered here since the high SNR is unphysical and the low SNR is more challenging. Many moving parts to problem - number of truncation rounds, the number of simulations per round, network architecture, the truncation, the sampling strategy of the priors.

Parameter	Prior dist.	Injection Value
Mass ratio, q	$U(0.125, 1)$	0.8858
Chirp mass, M [$M_\odot$]	$U(25, 100)$	32.14
Inclination angle, θ_{jn} [rad]	$\sin(0, \pi)$	0.4432
Phase, ϕ_c [rad]	$U(0, 2\pi)$	5.089
Tilt angle, θ_1 [rad]	$\sin(0, \pi)$	1.497
Tilt angle, θ_2 [rad]	$\sin(0, \pi)$	1.102
Spin, a_1	$U(0.05, 1)$	0.9702
Spin, a_2	$U(0.05, 1)$	0.8118
Spin angle, ϕ_{12} [rad]	$U(0, 2\pi)$	6.220
Spin angle, ϕ_{jl} [rad]	$U(0, 2\pi)$	1.885
Luminosity Distance, d_L [Mpc]	$U_{\text{vol}}(100, 2000)$	900
Right ascension, α [rad]	$U(0, 2\pi)$	5.556
Declination, δ [rad]	$\cos(-\pi/2, \pi/2)$	0.071
Polarisation angle, ψ [rad]	$U(0, \pi)$	1.100
Merger time, t_c [GPS s]	$U(-0.1, 0.1)$	0.000

Table 1: Description and type of the parameters that fully describe the detected gravitational waves. The injection values refer to the parameters used for the generated target observation, x_0.

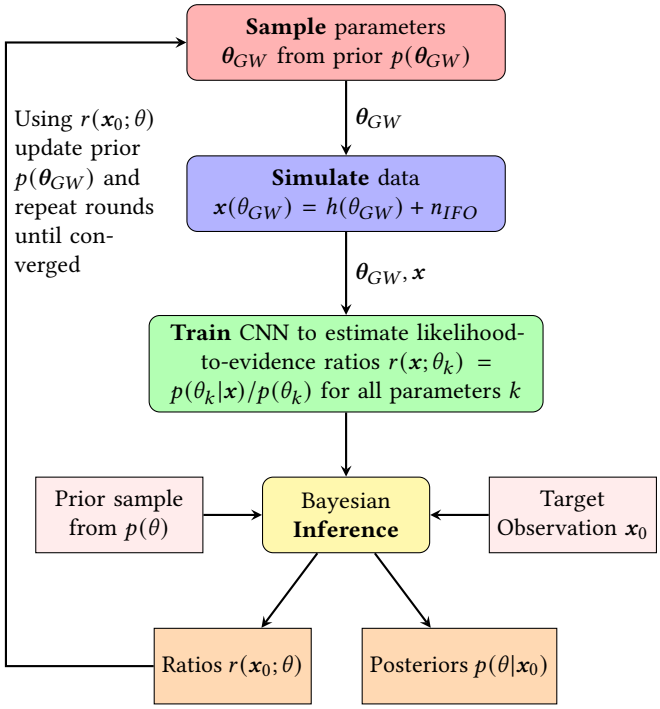


Figure 4: High-level overview of the simulation-based inference method used for this work.

4.3 Peregrine inference pipeline

The overall objective of this work is to increase the efficiency of the `Peregrine` data analysis pipeline. The work will begin with reproducing the results from papers [**?**] and [**?**], as this will form the benchmark to which the eventual results will be compared to.

The workflow for the simulation-based inference technique for the analysis of the gravitational wave signals as implemented in `peregrine` [**?**] is shown in Figure 4. The process starts by setting the 15 parameters of the 'target observation' and then generating the example waveform to be analysed. This is done so there are 'ground-truth' parameter values that you can compare your final posterior probability density distributions with and validate the overall method. If we use a true experimentally measured signal, then we can never know for certain what the 'ground-truth' values of the parameters are. Given the accuracy that we can forward model the GW signals with, once the method is validated with the simulated waveforms, it is expected to work equally well with true experimental measurements.

Fifteen individual binary classifiers. Minimise binary cross-entropy loss.

Noise shuffling.

Number of trainable parameters for each model.

The current rendition of the Peregrine pipeline requires 8 rounds of sequential TMNRE, 720000 simulated waveforms and around 12.5 hours (on a single A100 gpu with 18 cpu cores) to completely reconstruct the posteriors of the fifteen parameters. Of these 12.5 hours, around 10 is for training the network and 2.3 is for generating the waveforms used for training the network. Our primary focus is therefore the optimisation of the pipeline on the network itself, and as a secondary the number of simulated waveforms.

Trainer settings.

Joint and marginal pairs.

4.4 Overview of approach

We ran Peregrine with the following changes to the network architecture. For maximum consistency, all other settings remained the same.

4.5 Attention U-Net

The UNet modelled was modified to include the attention gate.

4.6 Transformer models

Two transformer architectures were considered, the so-called 1D vanilla vision transformer [**? ?**], and an architecture that was specifically designed for multi-variate time series representation learning [**?**].

4.6.1 Hyperparameter tuning. To find the optimal hyperparameters for each of the two models, we used the Python library Ray Tune citeliaw2018tune in combination with the Hyperopt Tree-structured Parzen Estimator search algorithm [**?**]. Ray Tune provides the framework to explore the hyperparameter space in a highly efficient way.

Ray Tune works by at first randomly selecting hyperparameters (in this case)

4.6.2 Pretraining. Transformers were pretrained for 24 hours with single A100 gpu on 2 million waveforms. After pretraining the saved weights were used to initialise the network during each Peregrine run. 95% train, 5% test. Regular validation checks of loss to ensure no overfitting and measure learning performance.

Table 2: Hyperparameters for ViT

Parameter	Values
batch_size	`tune.sample_from(lambda spec: spec.config.patch_size ** 2)`
learning_rate	`tune.loguniform(3e-5, 2e-4)`
patch_size	`tune.choice([4, 8, 16])`
num_classes	`tune.choice([16, 24, 32])`
dim	`tune.choice([256, 512, 1024])`
depth	`tune.randint(4, 10)`
heads	`tune.randint(4, 10)`
mlp_dim	`tune.choice([1024, 2048])`
dropout	`tune.choice([0, 0.05, 0.1])`
emb_dropout	`tune.choice([0, 0.05, 0.1])`
max_num_epochs	`max_num_epochs`

Table 3: Hyperparameters for tuning in mvts

Parameter
batch$_size = tune.choice([1]), learning_rate = tune.choice([1e-3, 3e-4, 1e-4]), d_model = tune.choice([128, 256, 512]), n_heads = tune.choice([4, 8, 16]), r$

Table 4: Loss values for each of the features

feature	mvts	vit	unet
mass$_ratio$	-0.231	-0.295	-0.300
chirp$_mass$	-0.542	-0.676	-0.701
theta$_jn$	-0.327	-0.463	-0.426
phase	0.000	0.000	0.000
tilt$_1$	-0.107	-0.167	-0.174
tilt$_2$	-0.018	-0.036	-0.036
a$_1$	-0.078	-0.104	-0.132
a$_2$	-0.005	-0.008	-0.007
phi$_12$	0.000	0.000	0.000
phi$_jl$	-0.199	-0.291	-0.287
luminosity$_distance$	-0.196	-0.294	-0.291
dec	-0.830	-0.923	-0.899
ra	-1.020	-1.085	-1.066
psi	-0.063	-0.117	-0.090
geocent$_time$	-0.965	-1.104	-1.105

4.7 Pruning

To reduce the size of the network we implemented some pruning methods with the DepGraph library [**?**].

4.8 Evaluation

5 RESULTS

5.1 Transformer models

In this section, we show the results during the pretraining of the transformer models

We can conclude that the ViT does not extract any additional features from the data, since the loss plateus to the same value. However, due to extra features the ViT is much slower. The mvts model is not a good model.

The pretrained ViT model was used for the evaluation of Peregrine.

AUROC for 15 features after 24 hours of training.

5.2 Peregrine Network

5.3 Peregrine Run Strategy

6 DISCUSSION

Write your discussion here. Do not forget to use sub-sections. Normally, the discussion starts with comparing your results to other studies as precisely as possible. The limitations should be reflected upon in terms such as reproducibility, scalability, generalizability, reliability and validity. It is also important to mention ethical concerns.

7 CONCLUSION

Write your conclusion here. Be sure that the relation between the research gap and your contribution is clear. Be honest about how limitations in the study qualify the answer on the research question.

Appendix A FIRST APPENDIX

Put your appendices here.