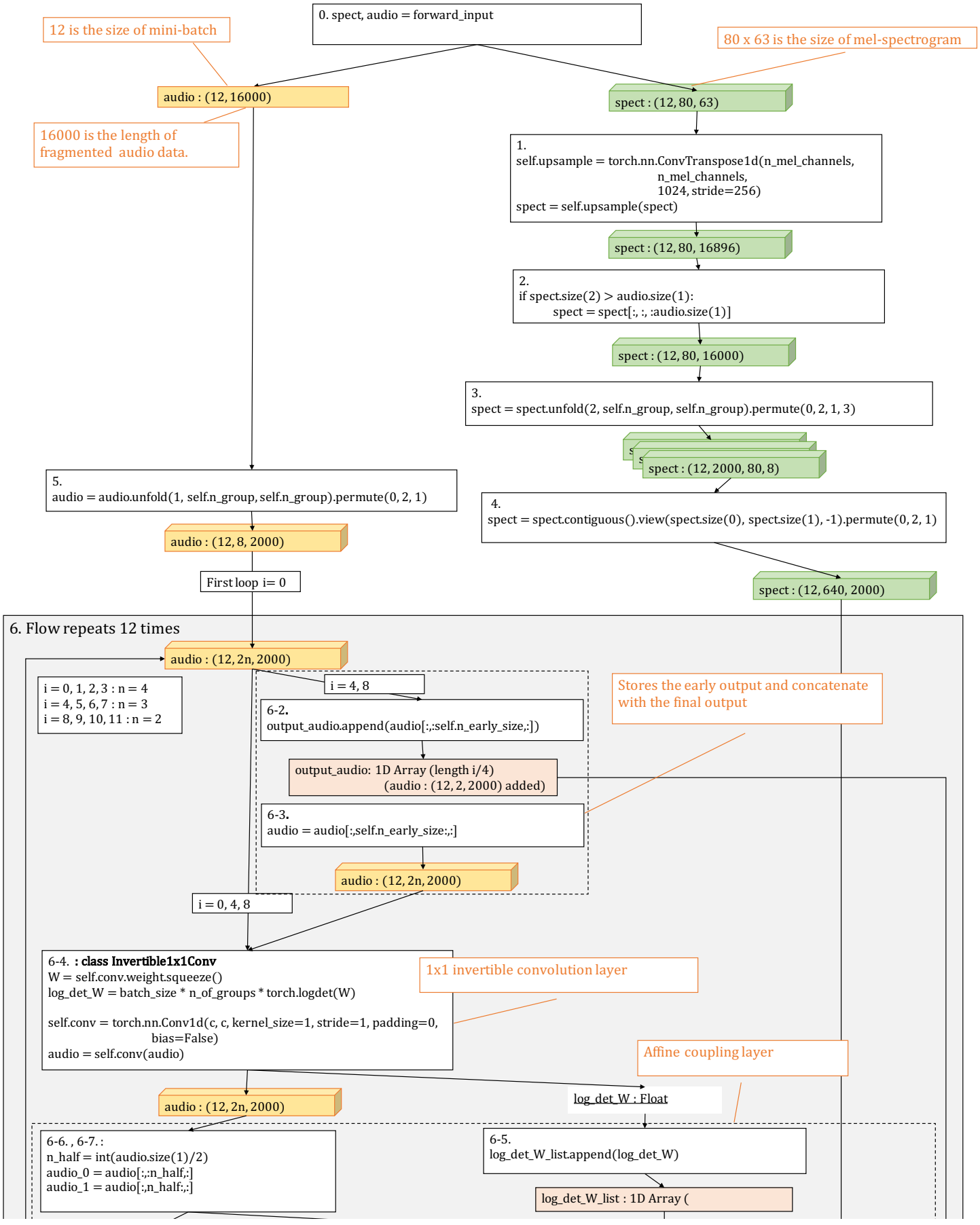
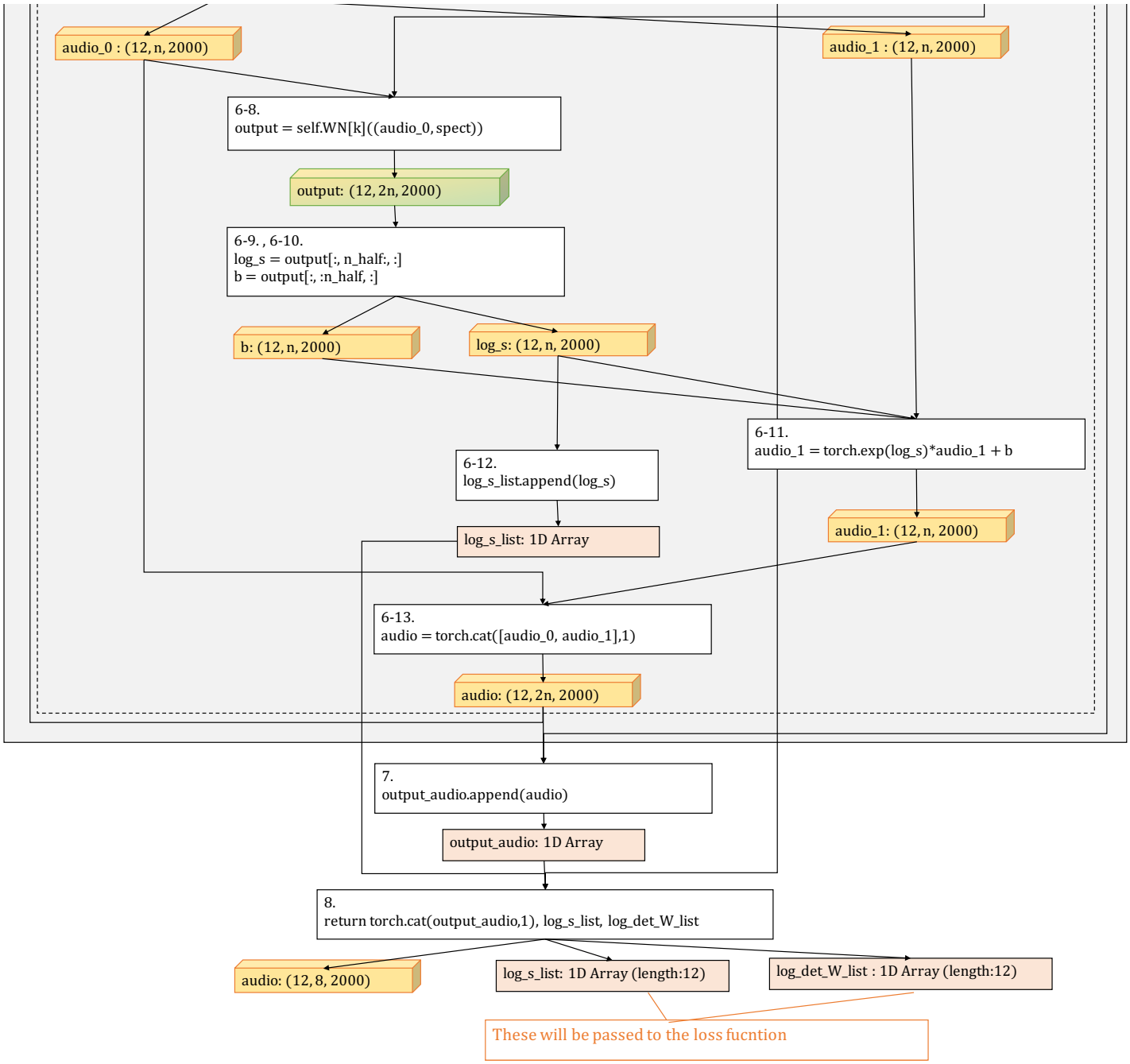
Forward pass

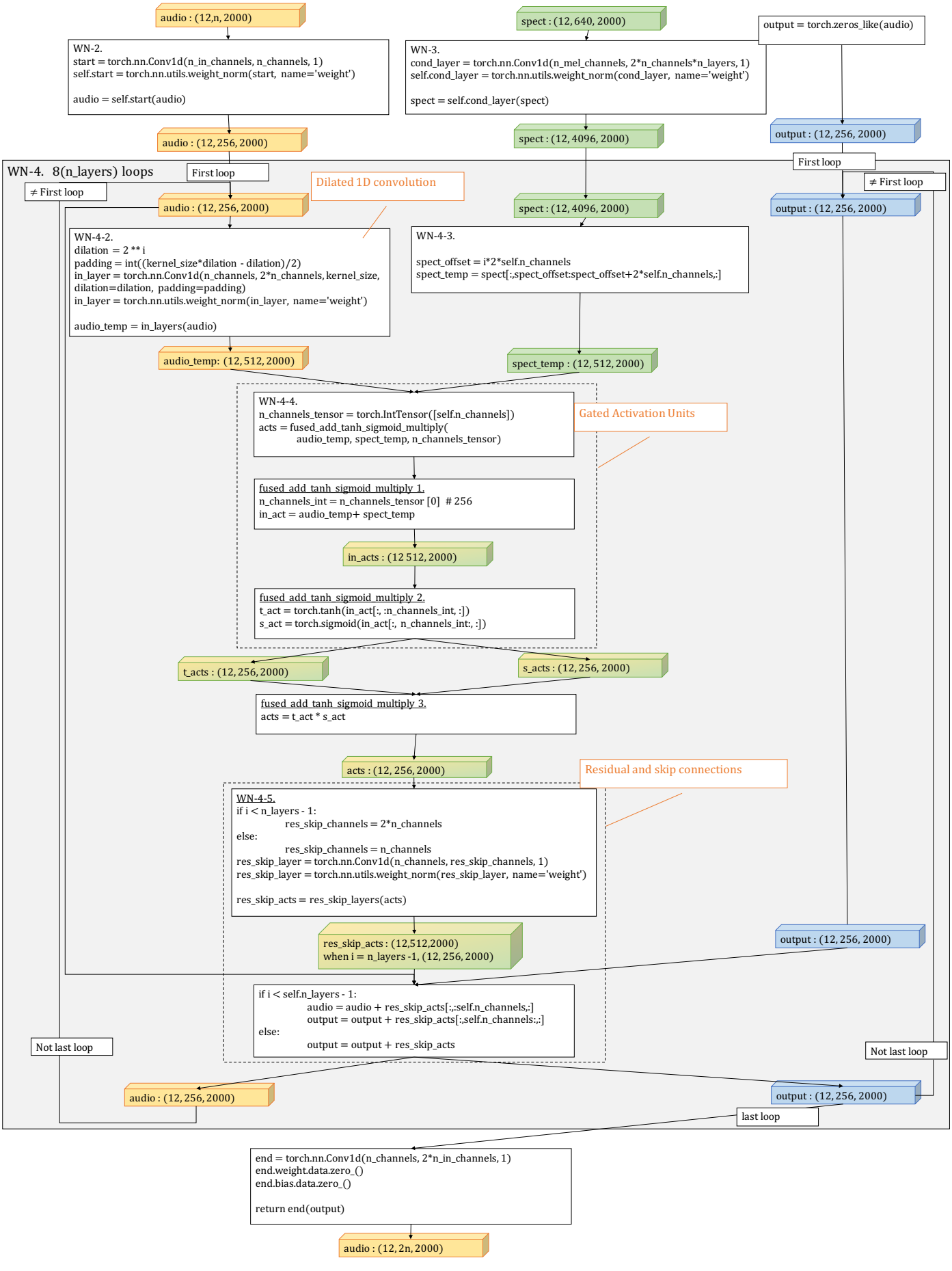
Data size is for the default configuration (https://github.com/NVIDIA/waveglow/blob/master/config.json)




audio_0 : (12, n, 2000)
audio_1 : (12, n, 2000)
6-8.
output = self.WN[k]((audio_0, spect))
output: (12, 2n, 2000)
6-9. , 6-10.
log_s = output[:, n_half:, :]
b = output[:, :n_half, :]
b: (12, n, 2000)
log_s: (12, n, 2000)
6-11.
audio_1 = torch.exp(log_s)*audio_1 + b
6-12.
log_s_list.append(log_s)
audio_1: (12, n, 2000)
log_s_list: 1D Array
6-13.
audio = torch.cat([audio_0, audio_1],1)
audio: (12, 2n, 2000)
7.
output_audio.append(audio)
output_audio: 1D Array
8.
return torch.cat(output_audio,1), log_s_list, log_det_W_list
audio: (12, 8, 2000)
log_s_list: 1D Array (length:12)
log_det_W_list : 1D Array (length:12)
These will be passed to the loss fucntion

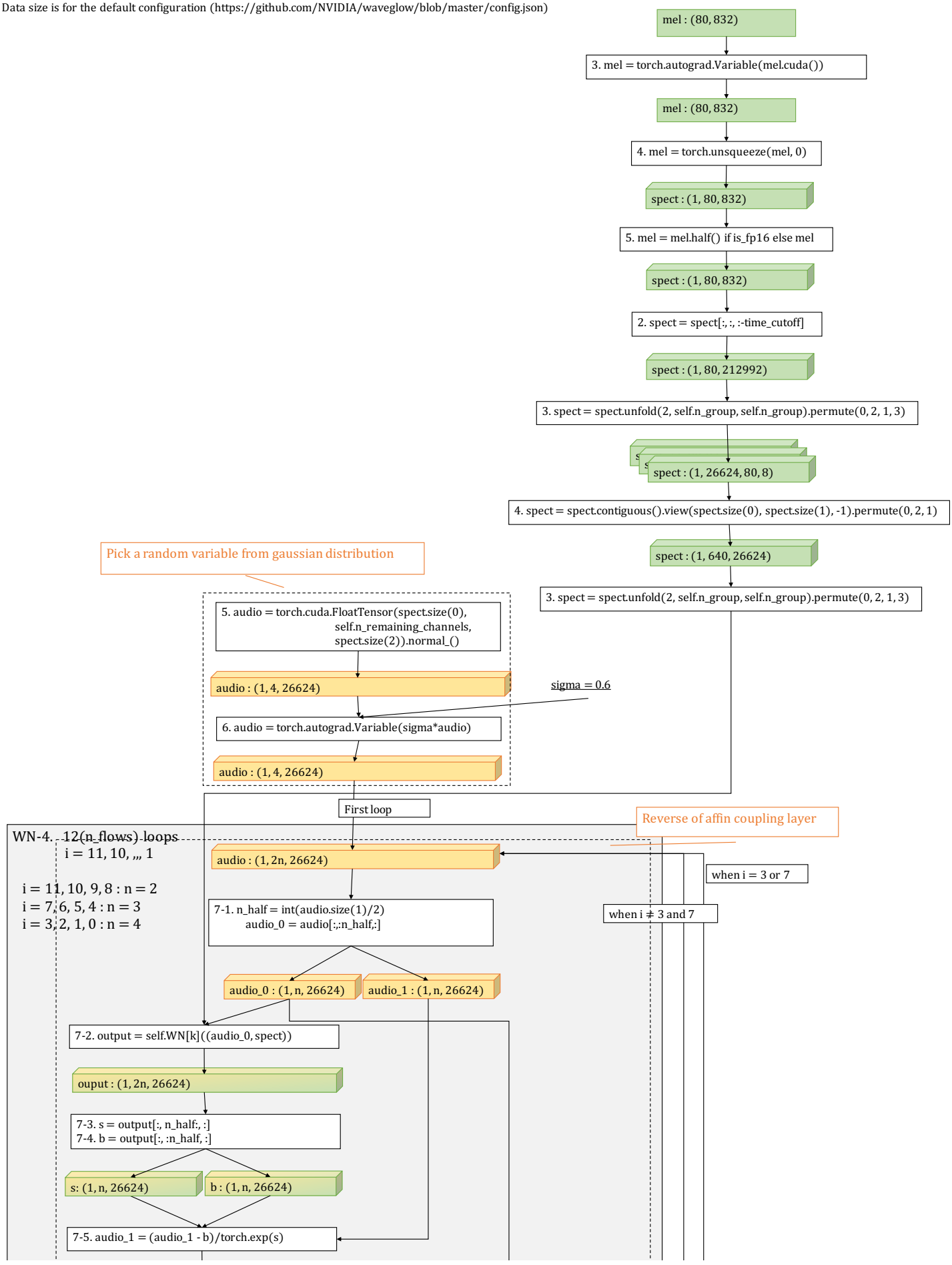
Coupling NN

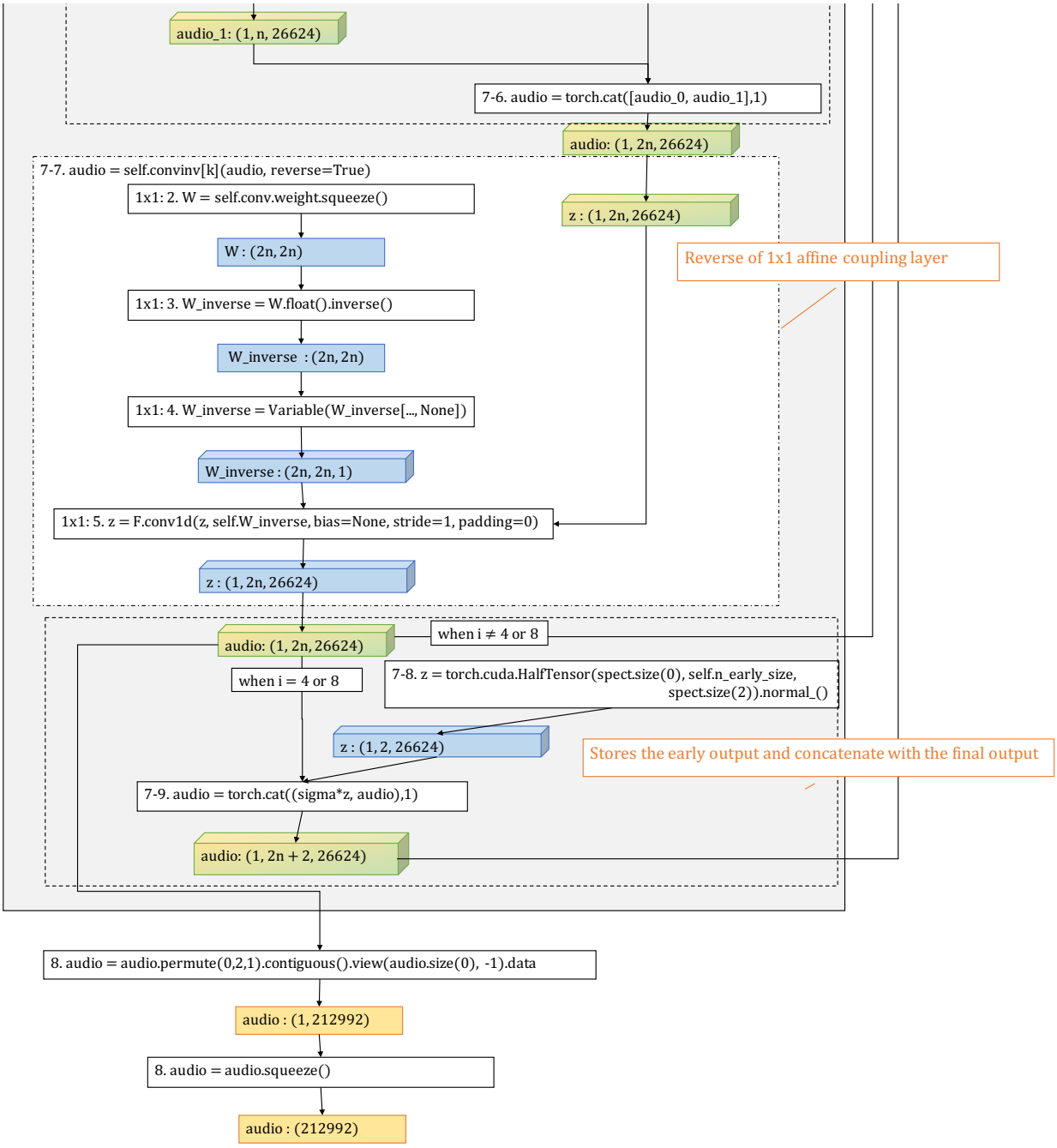
Data size is for the default configuration (https://github.com/NVIDIA/waveglow/blob/master/config.json)



Inference

Data size is for the default configuration (https://github.com/NVIDIA/waveglow/blob/master/config.json)




audio_1: (1, n, 26624)
7-6. audio = torch.cat([audio_0, audio_1],1)
audio: (1, 2n, 26624)
7-7. audio = self.convinv[k](audio, reverse=True)
1x1: 2. W = self.conv.weight.squeeze()
z : (1, 2n, 26624)
W : (2n, 2n)
Reverse of 1x1 affine coupling layer
1x1: 3. W_inverse = W.float().inverse()
W_inverse : (2n, 2n)
1x1: 4. W_inverse = Variable(W_inverse[..., None])
W_inverse : (2n, 2n, 1)
1x1: 5. z = F.conv1d(z, self.W_inverse, bias=None, stride=1, padding=0)
z : (1, 2n, 26624)
audio: (1, 2n, 26624)
when i ≠ 4 or 8
when i = 4 or 8
7-8. z = torch.cuda.HalfTensor(spect.size(0), self.n_early_size, spect.size(2)).normal_()
z : (1, 2, 26624)
Stores the early output and concatenate with the final output
7-9. audio = torch.cat((sigma*z, audio),1)
audio: (1, 2n + 2, 26624)
8. audio = audio.permute(0,2,1).contiguous().view(audio.size(0), -1).data
audio : (1, 212992)
8. audio = audio.squeeze()
audio : (212992)