MIDTERM 3 – ASSIGNMENT 1

Autoencoders

shallow – deep – contractive – denoising – convolutional - randomized

Pasquali Alex

Contractive

Subclassed Pytorch's MSELoss to use another function I created: contractive_loss

criterion = ContractiveLoss(...)

Use normally:

```
outputs = model(train_data_batch)
loss = criterion(outputs, train_targets_batch)
loss.backward()
optimizer.step()
```

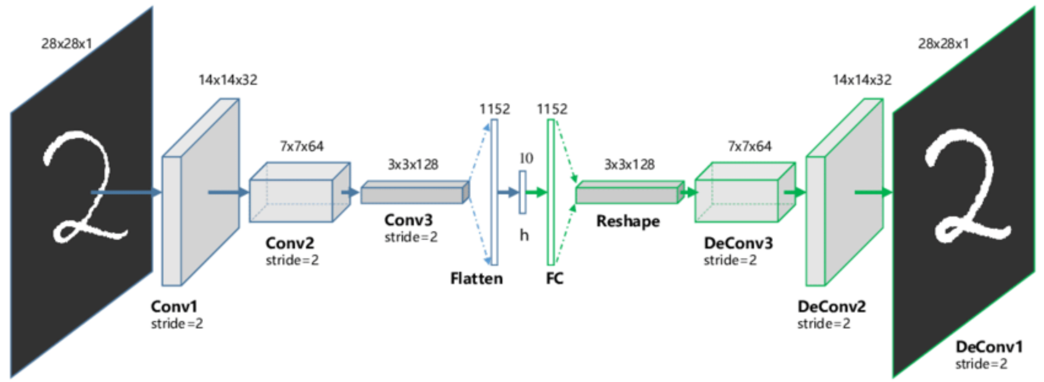
```
class ContractiveLoss(loss.MSELoss):
    """Custom loss for contractive autoencoders...."""
    def __init__(self, ae, lambd: float, size_average=None, reduce=None, reduction: str = 'mean') -> None:
        super(ContractiveLoss, self).__init__(size_average, reduce, reduction)
        self.ae = ae
        self.lambd = lambd

    def forward(self, input: Tensor, target: Tensor) -> Tensor:
        return contractive_loss(input, target, self.lambd, self.ae, self.reduction)


def contractive_loss(input, target, lambd, ae, reduction: str):
    """..."""
    term1 = (input - target) ** 2
    enc_weights = [ae.encoder[i].weight for i in reversed(range(1, len(ae.encoder), 2))]
    term2 = lambd * torch.norm(torch.chain_matmul(*enc_weights))
    contr_loss = torch.mean(term1 + term2, 0)
    if reduction == 'mean':
        return torch.mean(contr_loss)
    elif reduction == 'sum':
        return torch.sum(contr_loss)
    else:
        raise ValueError(f"value for 'reduction' must be 'mean' or 'sum', got {reduction}")
```

Denoising

Subclassed Pytorch's MNIST dataset to load all data in GPU memory at once, to speed up performance. Then create targets as a copy of data and add noise to the latter

```
class NoisyMNIST(FastMNIST):
    def __init__(self, noise_const=0.1, patch_width=0, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.data += noise_const * torch.randn(self.data.shape).to(device)
        self.data -= torch.min(self.data)
        self.data /= torch.max(self.data)
        if patch_width > 0:
            for img in self.data:
                start = random.randint(0, img.shape[-1] - patch_width - 1)
                img[:, start: start + patch_width, start: start + patch_width] = 0
```

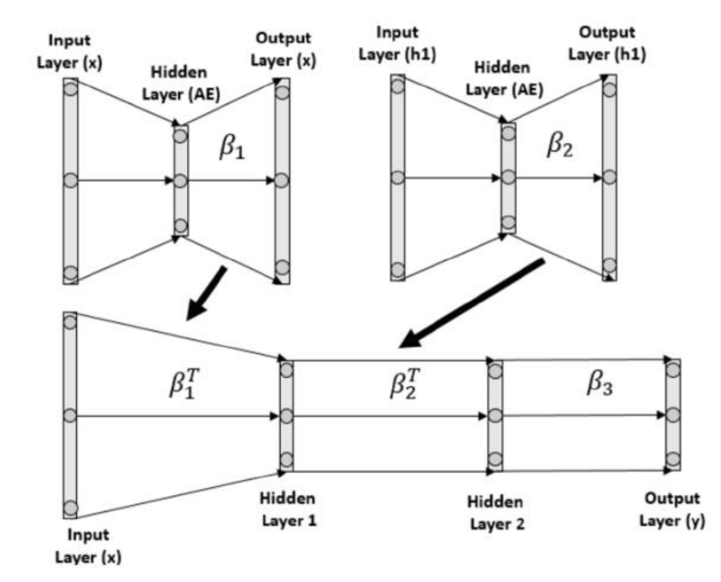
Convolutional



General architecture:

- Conv layers with ReLU: keep same area, increasing number of filters
- Max pooling (cause of the area reduction)
- Fully connected layer in the center
- Decoder: deconvolutions with decreasing number of filters to restore the original dimension

Randomized

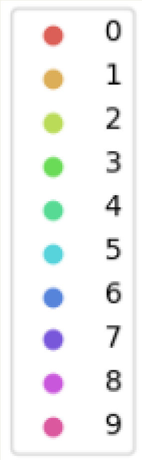


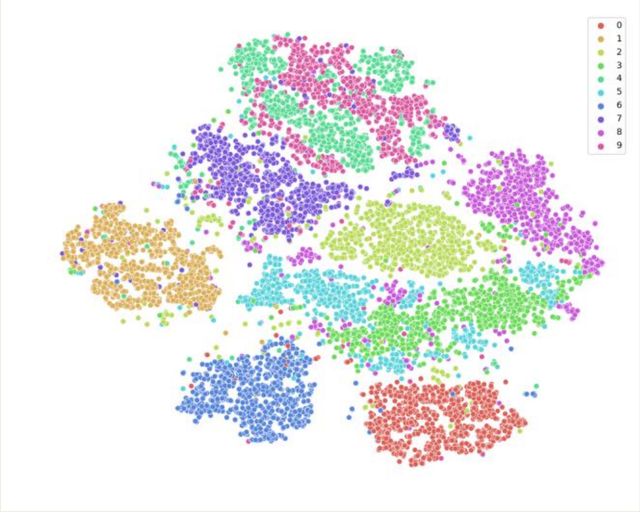
For each layer of your deep randomized autoencoder:

- Create a shallow AE with the latent dimension equal to the current layer's one
- Randomly initialize the weight matrices of encoder and decoder
- Train shallow AE keeping **encoder's weights fixed**
- Copy decoder's weights W_d in the correct layer of the **deep *decoder***
- Copy W_d **transpose** in the correct layer of the **deep *encoder***

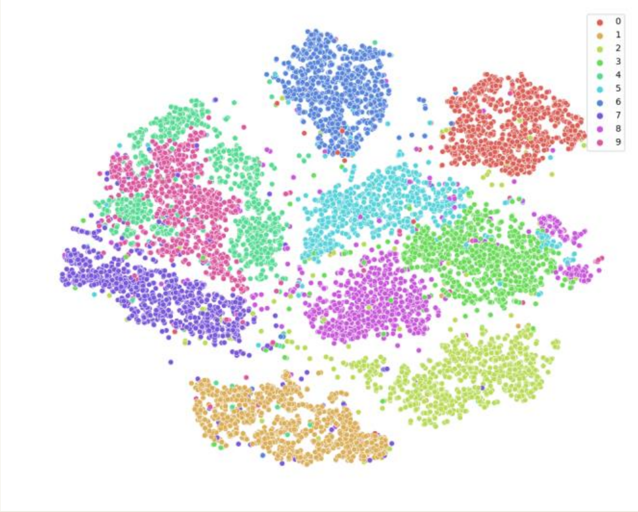
Std AEs	Basic	Contractive	Denoising

Shallow

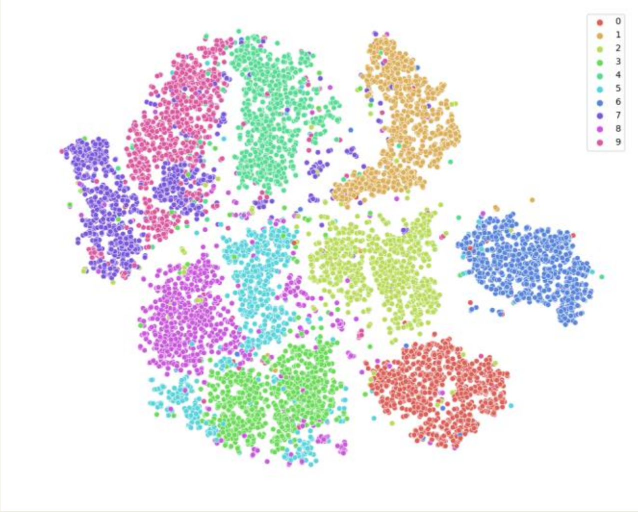




Classification accuracy: 91,03%

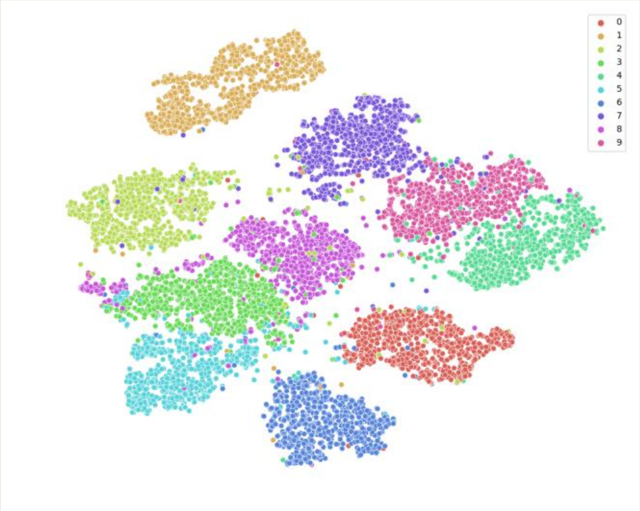


Classification accuracy: 91,49%

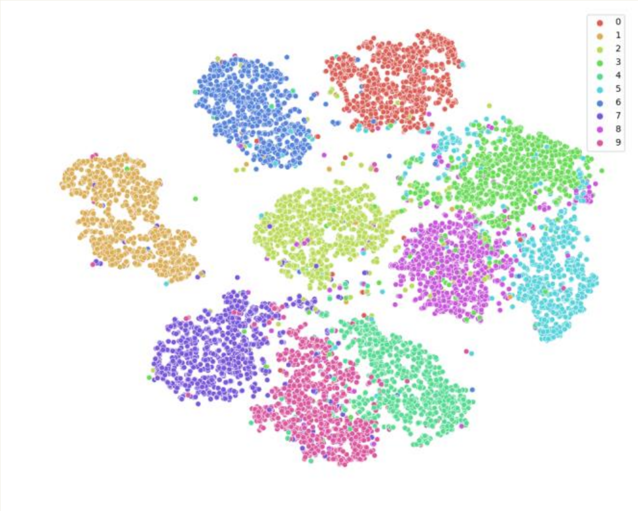


Classification accuracy: 91,89%

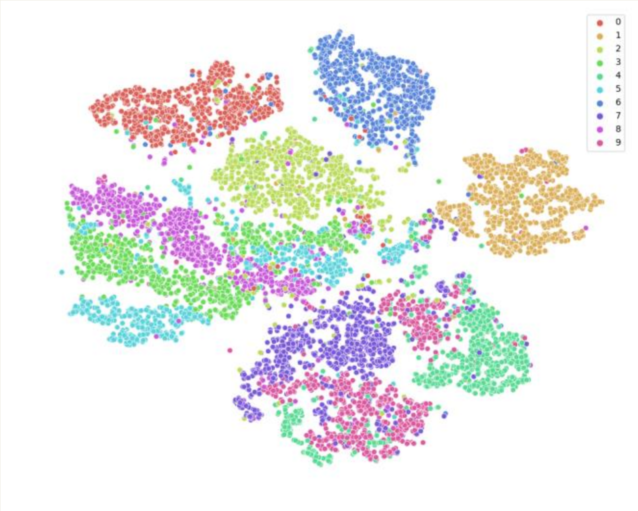
Deep



Classification accuracy: 90,02%



Classification accuracy: 87,57%



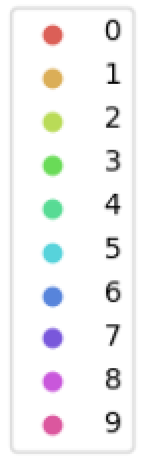
Classification accuracy: 88,57%

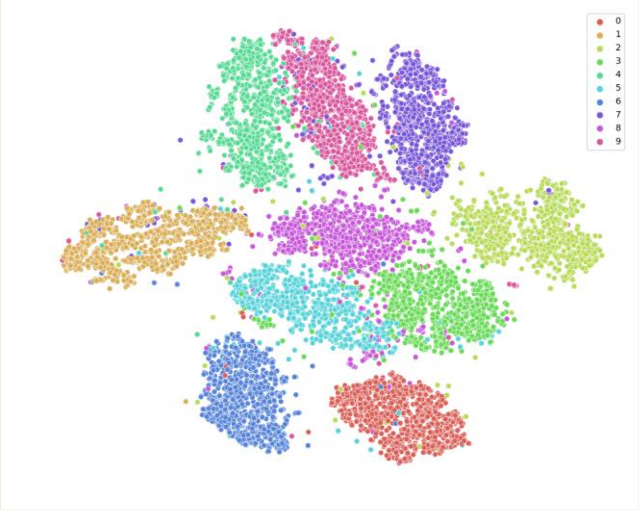
Shallow Conv

Deep Conv

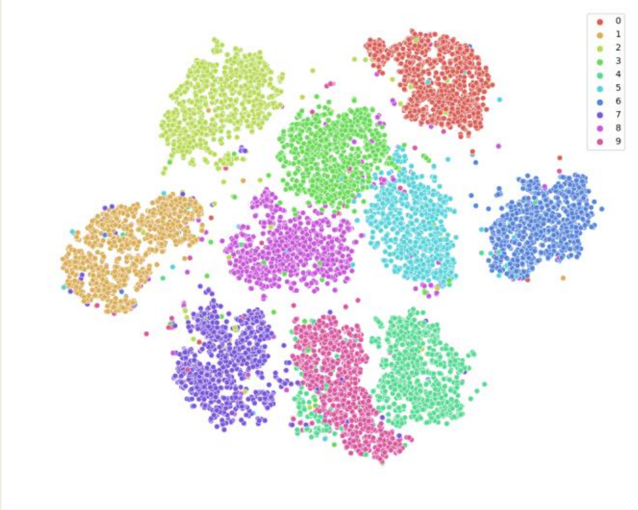
Randomized

Basic

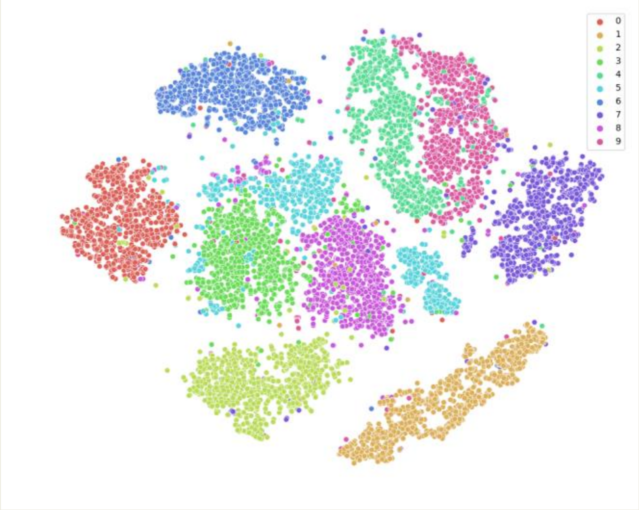


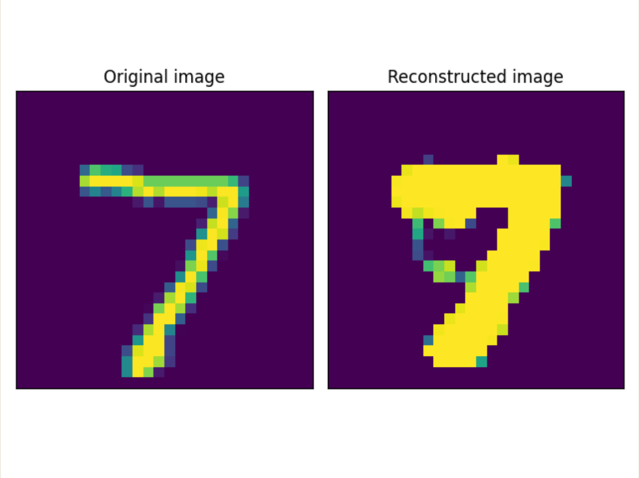


Classification accuracy: 91,11%



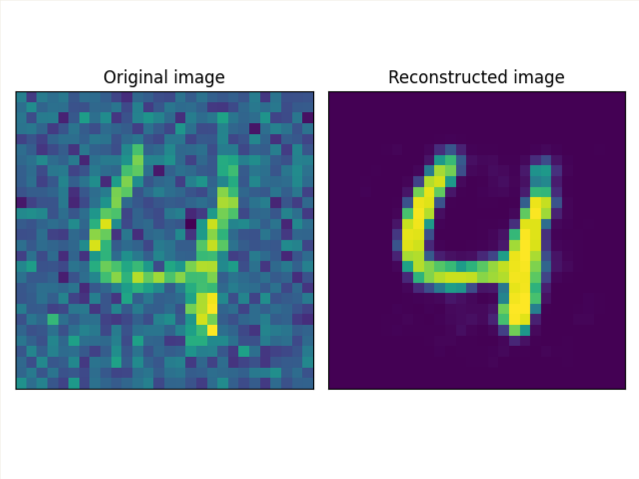
Classification accuracy: 92,73%



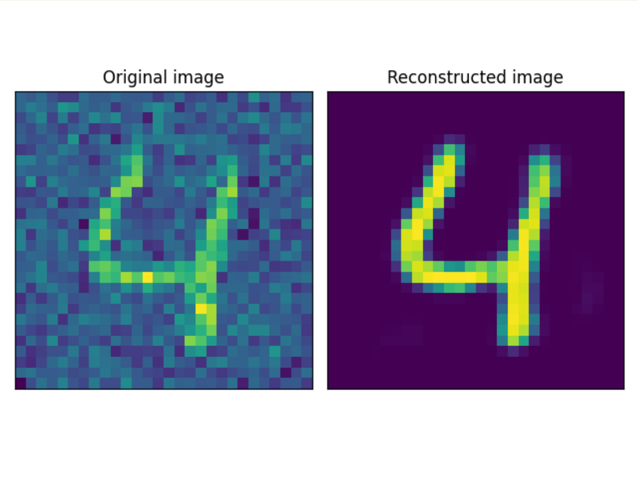


Classification accuracy: 88,46%

Denoising



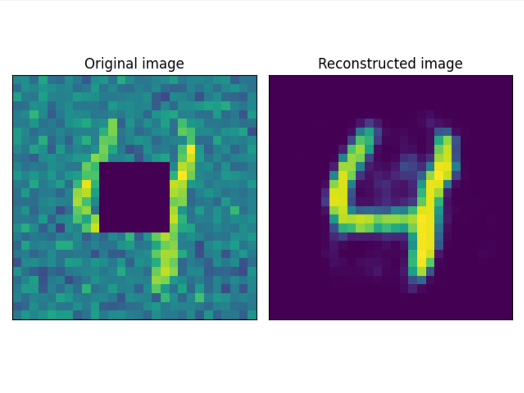
Classification accuracy: 91,77%

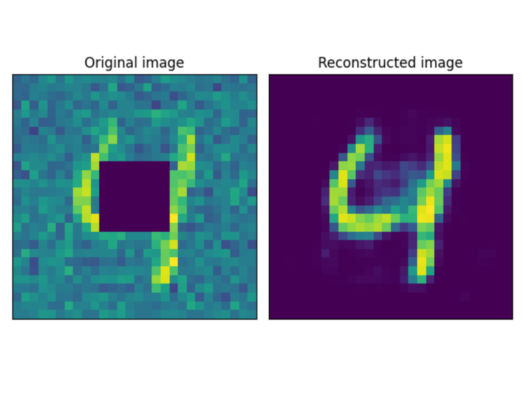


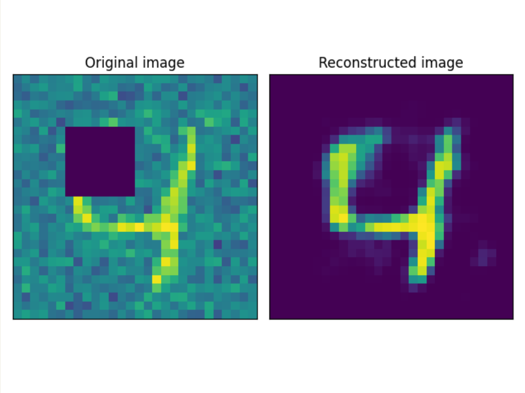
Classification accuracy: 93,60%

Image occlusion

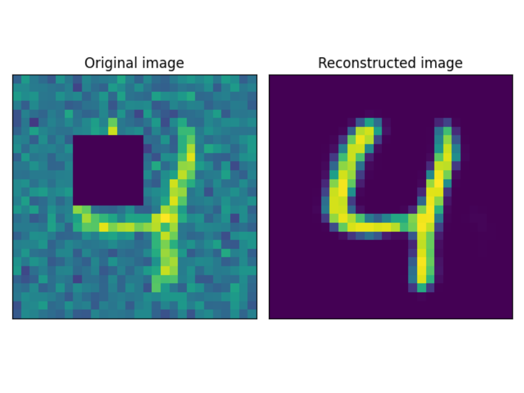
Shallow Conv

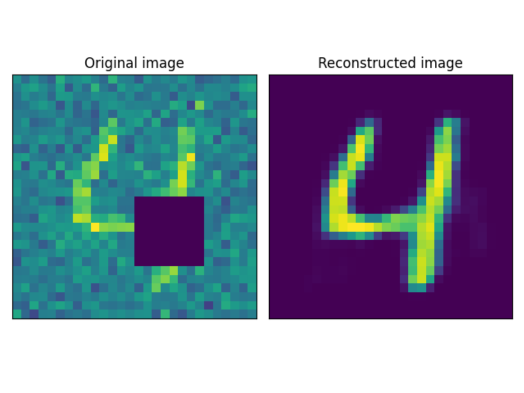


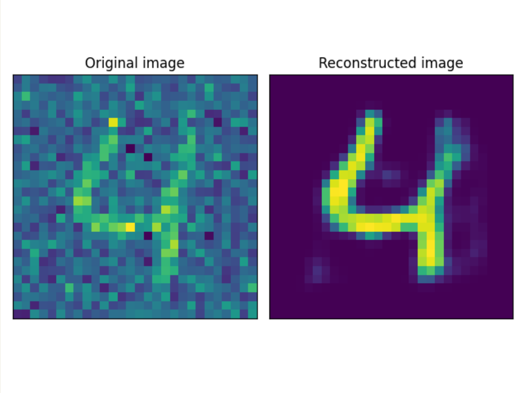




Deep Conv





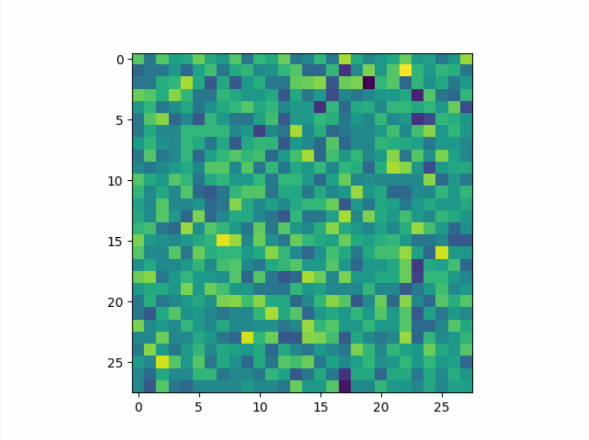


Manifold convergence

Better visualization if AE has smaller latent dimensionality.

This case:

- deep contractive AE
- (784, 500, 200, 100, 10)



Few observations

- t-SNE plots and classification accuracy are pretty good, regardless of the models
- Convolutional AE are only slightly better than standard AE
  - Probably due to the simplicity of the images in MNIST
  - However they reach good results way faster than standard AE (less epochs)
  - Still, they need more time to perform well on image occlusion

Possible future works

- Wider training and testing of randomized AE
  - With noisy images
  - With image occlusion
- Use denoising AE for image colorization
- Echo State Networks for MNIST recognition (Schaetti et al., 2016)

THANK YOU FOR YOUR ATTENTION

Pasquali Alex