```
import torch
from transformers import AutoTokenizer, AutoModel
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import numpy as np
import sympy as sp

# Load ProtBert model from HuggingFace
tokenizer = AutoTokenizer.from_pretrained("Rostlab/prot_bert", do_lower_case=False)
model = AutoModel.from_pretrained("Rostlab/prot_bert")

analyzer = SentimentIntensityAnalyzer()

def fuse_perspectives(target_signature, models=['newton', 'davinci', 'quantum', 'ethics']):
    sequence = target_signature['cleaned_sequence']
    encoded_input = tokenizer(sequence, return_tensors="pt")
    with torch.no_grad():
        embedding = model(**encoded_input).last_hidden_state.mean(dim=1).squeeze().numpy()

    # Normalize vector
    norm_embedding = embedding / np.linalg.norm(embedding)

    # Simulated reasoning output
    sentiment = analyzer.polarity_scores(sequence)
    symbolic_logic = sp.sympify(target_signature['isoelectric_point']) + sp.Rational(1, 3)
```

```
fused_output = {
    "embedding_vector": norm_embedding.tolist(),
    "sentiment_trace": sentiment,
    "symbolic_logic_score": float(symbolic_logic),
    "perspective_tags": models,
    "reasoning_fusion": "Completed"
}

return fused_output
```